

# Modeling Distributed Private Key Generation by Composing Petri Nets

Luca Bernardinello<sup>1(✉)</sup>, Görkem Kılınc<sup>1</sup>,  
Elisabetta Mangioni<sup>1,2</sup>, and Lucia Pomello<sup>1</sup>

<sup>1</sup> Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli studi di Milano - Bicocca,  
Viale Sarca, 336 - Edificio U14, 20126 Milano, Italy  
bernardinello@disco.unimib.it

<sup>2</sup> Istituto per la Dinamica dei Processi Ambientali,  
Consiglio Nazionale delle Ricerche (CNR-IDPA),  
Piazza della Scienza, 1 - Edificio U1, 20126 Milano, Italy

**Abstract.** We present a Petri net model of a protocol for the distributed generation of id-based private keys. Those keys can then be used for secure communications. The components of the system are built as refinements of a common interface, by applying a formal operation based on a class of morphisms between Elementary Net Systems. It is then shown that we can derive behavioural properties of the composed system without building it explicitly, by exploiting properties of the given morphisms.

**Keywords:** Petri nets · Morphisms · Local state refinement · Composition · Distributed private key generation

## 1 Introduction

In [1] we proposed a way to compose Elementary Net (EN) Systems by identifying conditions, places, and events. The identification is ruled by a pair of morphisms from the two components to an *interface*. The interface is an EN System which can be seen as specifying the protocol of interaction between components, or a common abstraction.

This framework was first defined relying on N-morphisms, originally introduced in [2, 9]. Later, the same operation was defined over a new class of morphisms, called  $\alpha$ -morphisms (see [1]).

An  $\alpha$ -morphism from an EN System  $N_1$  to an EN System  $N_2$  corresponds to a relation of refinement: some subnets of  $N_1$  refine conditions of  $N_2$ . This refinement may require that some events be duplicated. Such morphisms are defined and discussed in Sect. 3.

When composing two EN Systems,  $N_1$  and  $N_2$ , over an interface  $N_I$ , the two morphisms towards the interface specify how each component refines parts of the interface. An uninterpreted example is given in Sect. 4.

One of the claimed advantages of this approach to design is the ability to derive properties of the composed systems from properties of the components and of the morphisms, without the need to actually build and analyze the composed system.

Ideally, one would like to derive behavioral properties, like liveness and safety properties, by analyzing only the structure of the models involved, thus avoiding the potentially high cost of computing the reachable markings. This is not always possible. Hence, the method we propose uses some behavioral information about components and interface; however, this is limited to only a part of the models, and does not involve the whole system model.

Another significant advantage of  $\alpha$ -morphisms is that model checking becomes easier provided that some additional requirements are satisfied. More detailed explanation is given in Sect. 3 and a model checking example is sketched in Sect. 6.

Here, we test these ideas on a protocol for distributed generation of id-based cryptographic keys. The protocol, described in more detail in Sect. 5, requires the cooperation of several *private key generators* (PKG) so that a client can build a private key. Basically,  $n$  PKG nodes come together to generate a master key pair consisting of a private and a public key. After that, each PKG node has a share for the master key pair. A client who wants to have a private key applies to  $k$  available PKG nodes. Each PKG node calculates a piece of the client's private key by using the unique id-string of the client and the share of the master private key which is held by that specific PKG node. On receiving  $k$  pieces, the client continues to extract its private key. The so called bulletin board is responsible for the initialization of the components in the system and broadcasting the public parameters. During both the distributed generation of the master key pair and the extraction of the clients' private keys, a verification can be performed by using the commitment values and public keys held and broadcast by the bulletin board. The id-based distributed private key generation protocol was proposed in [3]; an improved version is presented in [4]. In [5], a Petri net model for the protocol to be implemented on industrial control systems is presented.

In the next section, basic definitions related to EN Systems are recalled. Section 3 recalls the formal definition of  $\alpha$ -morphisms and the properties they preserve or reflect and which are used in the rest of the paper. The definition of an operation of composition of EN Systems, based on  $\alpha$ -morphisms, is informally recalled in Sect. 4 on the basis of an uninterpreted example. In the same section, the main result relating behavioral properties of the composed system to behavioral properties of its components is recalled. Section 5 presents the distributed private key generation protocol which is modeled by Petri nets in Sect. 6. In the same section, we analyze behavioral properties of the model. The paper is closed by a short concluding section.

## 2 Preliminary Definitions

In this section, we recall the basic definitions of net theory, in particular Elementary Net Systems and unfoldings [11].

A *net* is a triple  $N = (B, E, F)$ , where  $B$  is a set of *conditions* or local states,  $E$  is a set of *events* or transitions such that  $B \cap E = \emptyset$ , and  $F \subseteq (B \times E) \cup (E \times B)$  is the *flow relation*.

We adopt the usual graphical notation: conditions are represented by circles, events by boxes and the flow relation by arcs. The set of elements of a net will be denoted by  $X = B \cup E$ .

The *preset* of an element  $x \in X$  is  $\bullet x = \{y \in X \mid (y, x) \in F\}$ ; the *postset* of  $x$  is  $x^\bullet = \{y \in X \mid (x, y) \in F\}$ ; the *neighborhood* of  $x$  is given by  $\bullet x^\bullet = \bullet x \cup x^\bullet$ . These notations are extended to subsets of elements in the usual way: i.e., let  $A \subseteq X$ ,  $\bullet A = \bigcup_{x \in A} \bullet x$ .

For any net  $N$  we denote the *in-elements* of  $N$  by  ${}^\circ N = \{x \in X : \bullet x = \emptyset\}$  and the *out-elements* of  $N$  by  $N^\circ = \{x \in X : x^\bullet = \emptyset\}$ . When in-elements or out-elements are only conditions, we call them respectively, *in-conditions* or *out-conditions*.

A net  $N' = (B', E', F')$  is a *subnet* of  $N = (B, E, F)$  if  $B' \subseteq B$ ,  $E' \subseteq E$ , and  $F' = F \cap ((B' \times E') \cup (E' \times B'))$ . Given a subset of elements  $A \subseteq X$ , we say that  $N(A)$  is the *subnet of  $N$  identified by  $A$*  if  $N(A) = (B \cap A, E \cap A, F \cap (A \times A))$ .

A *State Machine* is a connected net such that each event  $e$  has exactly one input condition and exactly one output condition:  $\forall e \in E, |\bullet e| = |e^\bullet| = 1$ .

Elementary Net (EN) Systems are a basic system model in net theory. An *Elementary Net System* is a quadruple  $N = (B, E, F, m_0)$ , where  $(B, E, F)$  is a net such that  $B$  and  $E$  are finite sets, self-loops are not allowed,  $\forall e \in E : \bullet e \cap e^\bullet = \emptyset$ , isolated elements are not allowed, and the *initial marking* is  $m_0 \subseteq B$ .

A subnet of an EN System  $N$  identified by a subset of conditions  $A$  and all its pre- and post-events,  $N(A \cup \bullet A^\bullet)$ , is a *Sequential Component* of  $N$  if  $N(A \cup \bullet A^\bullet)$  is a State Machine and if it has only one token in the initial marking.

An EN System is *covered* by Sequential Components if every condition of the net belongs to at least a Sequential Component. In this case we say that the system is *State Machine Decomposable (SMD)*.

Let  $N = (B, E, F, m_0)$  be an EN System,  $e \in E$  and  $m \subseteq B$ . The event  $e$  is *enabled* at  $m$ , denoted  $m[e]$ , if  $\bullet e \subseteq m$  and  $e^\bullet \cap m = \emptyset$ ; the occurrence of  $e$  at  $m$  leads from  $m$  to  $m'$ , denoted  $m[e]m'$ , iff  $m' = (m \setminus \bullet e) \cup e^\bullet$ . We also say that  $e$  fires at  $m$ .

Let  $\epsilon$  denote the empty word in  $E^*$ . The firing rule is extended to sequences of events by setting  $m[\epsilon]m$  and  $\forall e \in E, \forall w \in E^*, m[ew]m' = m[e]m''[w]m'$ ;  $w$  is called *firing sequence*.

A subset  $m \subseteq B$  is a *reachable marking* of  $N$  if there exists a  $w \in E^*$  such that  $m_0[w]m$ . The *set of all reachable markings* of  $N$  is denoted by  $[m_0]$ .

An EN System is *contact-free* if  $\forall e \in E, \forall m \in [m_0] : \bullet e \subseteq m$  implies  $e^\bullet \cap m = \emptyset$ . An EN System covered by Sequential Components is contact-free [11]. An event is called *dead* at a marking  $m$  if it is not enabled at any marking reachable from  $m$ . A reachable marking  $m$  is called *dead* if no event is enabled at  $m$ . An EN System is *deadlock-free* if no reachable marking is dead.

The semantics of an EN System can be given as its *unfolding*. The unfolding is an acyclic net, possibly infinite, which records the occurrences of its elements in all possible executions.

In order to define occurrence nets, firstly we need to define the relation  $\#$ . Let  $N = (B, E, F)$  be a net, and let  $x, y \in X$ . We say that  $x$  and  $y$  are in  $\#$  relation, if there exist two distinct events  $e_x, e_y \in E$  such that  $e_x F^* x$ ,  $e_y F^* y$ , and  $\bullet e_x \cap \bullet e_y \neq \emptyset$ , where  $F^*$  is the reflexive and transitive closure of  $F$ .

An *occurrence net* is a net  $N = (B, E, F)$  such that if  $e_1, e_2 \in E$ ,  $e_1 \bullet \cap e_2 \bullet \neq \emptyset$  then  $e_1 = e_2$ ;  $F^*$  is a partial order; for any  $x \in X$ ,  $\{y : y F^* x\}$  is finite;  $\#_N$  is irreflexive and the minimal elements with respect to  $F^*$  are conditions. Occurrence nets were introduced in [8, 11].

A branching process of an EN System  $N$  is an occurrence net whose elements can be mapped to the elements of  $N$ . Let  $N = (B, E, F, m_0)$  be an EN System, and  $\Sigma = (P, T, G)$  be an occurrence net. Let  $\pi : P \cup T \rightarrow B \cup E$  be a total map. The pair  $(\Sigma, \pi)$  is a *branching process* of  $N$  if  $\pi(P) \subseteq B$ ,  $\pi(T) \subseteq E$ ;  $\pi$  restricted to the minimal elements of  $\Sigma$  is a bijection on  $m_0$ ; for each  $t \in T$ ,  $\pi$  restricted to  $\bullet t$  is injective and  $\pi$  restricted to  $t \bullet$  is injective and for each  $t \in T$ ,  $\pi(\bullet t) = \bullet(\pi(t))$  and  $\pi(t \bullet) = (\pi(t)) \bullet$ .

The *unfolding* of an EN System  $N$ , denoted by  $Unf(N)$ , is the maximal branching process of  $N$ , namely the unique, up to isomorphism, branching process such that any other branching process of  $N$  is isomorphic to a subnet of  $Unf(N)$ . The map associated to the unfolding will be denoted  $u$  and called *folding*.

Bisimulation relations have been introduced as equivalence notions with respect to event observation [7]. The observability of events of a system is defined by using a labelling function which associates the same label to different events, when viewed as equal by an observer, and the label  $\tau$  to unobservable events. Let  $N = (B, E, F, m_0)$  be an EN System,  $l : E \rightarrow L \cup \{\tau\}$  be a labelling function where  $L$  is the alphabet of observable actions and  $\tau \notin L$  the unobservable action. Let  $\epsilon$  denote the empty word both of  $E^*$  and  $L^*$ . The function  $l$  is extended to a homomorphism  $l : E^* \rightarrow L^*$  in the following way:

$$l(\epsilon) = \epsilon$$

$$\forall e \in E, \forall w \in E^*, l(ew) = \begin{cases} l(e)l(w) & \text{if } l(e) \neq \tau \\ l(w) & \text{if } l(e) = \tau \end{cases}$$

The pair  $(N, l)$  is called *Labelled EN System*.

Let  $m, m' \in [m_0]$  and  $a \in L \cup \{\epsilon\}$ ; then:

- $a$  is enabled at  $m$ , denoted  $m(a)$ , iff  $\exists w \in E^* : l(w) = a$  and  $m[w]$ ;
- if  $a$  is enabled at  $m$ , then the occurrence of  $a$  can lead from  $m$  to  $m'$ , denoted  $m(a) m'$ , iff  $\exists w \in E^* : l(w) = a$  and  $m[w] m'$ .

Weak bisimulation is defined as a relation between reachable markings of Labelled EN Systems [10].

Let  $N_i = (B_i, E_i, F_i, m_0^i)$  be an EN System for  $i = 1, 2$ , with the labelling function  $l_i : E_i \rightarrow L \cup \{\tau\}$ . Then  $(N_1, l_1)$  and  $(N_2, l_2)$  are *weakly bisimilar*, denoted  $(N_1, l_1) \approx (N_2, l_2)$ , iff  $\exists r \subseteq [m_0^1] \times [m_0^2]$  such that:

- $(m_0^1, m_0^2) \in r$ ;
- $\forall (m_1, m_2) \in r, \forall a \in L \cup \{\epsilon\}$  it holds

$$\forall m'_1 : m_1(a) m'_1 \Rightarrow \exists m'_2 : m_2(a) m'_2 \wedge (m'_1, m'_2) \in r$$

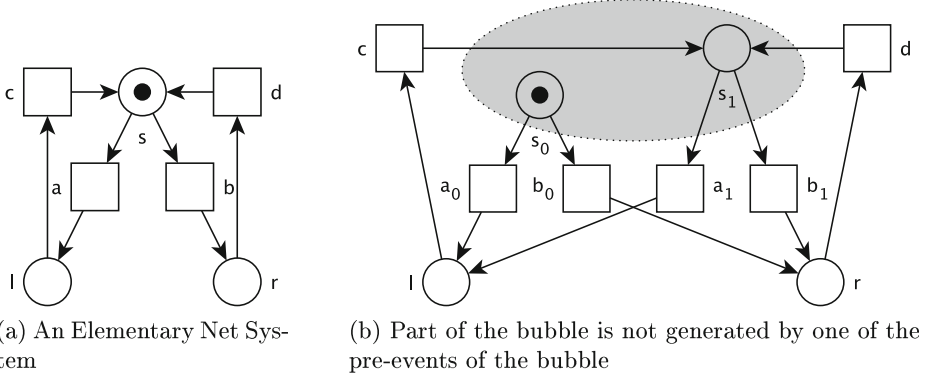
and (vice versa)

$$\forall m'_2 : m_2(a) m'_2 \Rightarrow \exists m'_1 : m_1(a) m'_1 \wedge (m'_1, m'_2) \in r$$

Such a relation  $r$  is called *weak bisimulation*.

### 3 $\alpha$ -morphisms

In this section we present the formal definition of  $\alpha$ -morphisms for State Machine Decomposable Elementary Net Systems (SMD-EN Systems) and the structural and behavioral properties that  $\alpha$ -morphisms preserve and reflect. For a comparison of  $\alpha$ -morphisms with other notions introduced in the literature, see [1].



**Fig. 1.** A net and one of its refinements

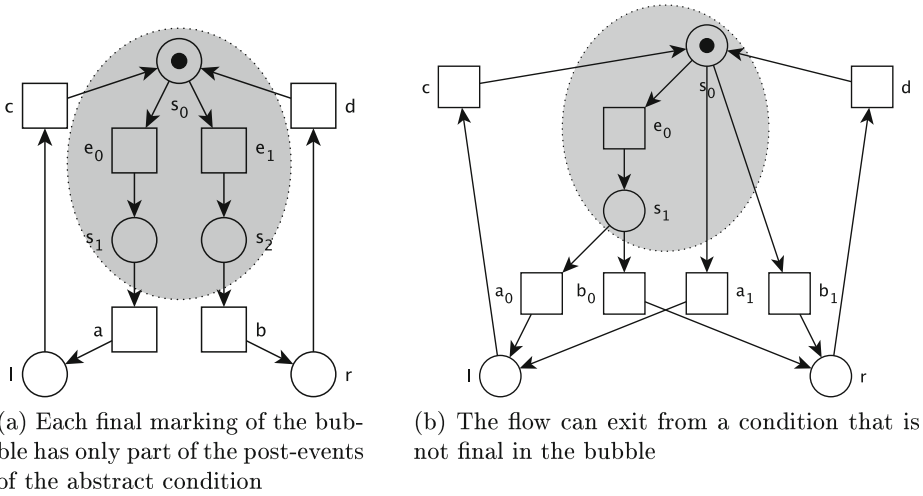
The proposed approach is motivated by the attempt to define a refinement operation preserving behavioural properties on the basis of structural and only local behavioural constraints. The additional restrictions, with respect to general morphisms, aim, on one hand, to capture typical features of refinements, and on the other hand to ensure that some behavioural properties of the abstract model still hold in the refined model.

An  $\alpha$ -morphism can map a subnet to a single place, which is then considered as an abstraction of that subnet. In this case, the subnet will be called a *bubble*. So, in general, the existence of an  $\alpha$ -morphism from  $N_1$  to  $N_2$  means that  $N_1$  can

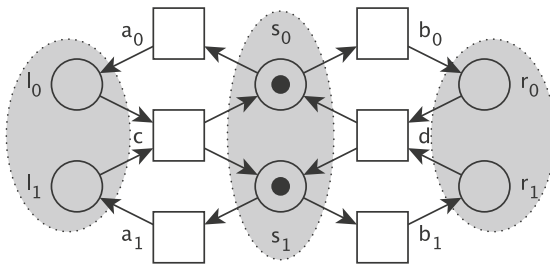
be seen as a refinement of  $N_2$ , where some places are replaced by more detailed subnets, or, viceversa, that  $N_2$  is an abstraction of  $N_1$ .

The formal definition of  $\alpha$ -morphism, given in this section, puts some restrictions on the structure and the behaviour of subnets mapped to a single place, and on the way these subnets are linked to the rest of the net. These restrictions are informally explained below, with the help of simple examples. The rest of the section comprises the related formal definitions, and deals with behavioural properties which are preserved or reflected by  $\alpha$ -morphisms.

Regarding the restrictions mentioned above, we require that a bubble does not contain an initialization part; in Fig. 1b we can see a refinement of the net of Fig. 1a in which the bubble contains an initialization part that will be executed only once. Moreover, each final marking of the bubble must have all the possibilities of the corresponding abstract condition (for a counterexample



**Fig. 2.** Two refinements of the net of Fig. 1a



**Fig. 3.** A refinement of the net of Fig. 1a in which no sequential component contains all the pre- and post-events of the bubble of  $s$

see Fig. 2a). We also do not want that a token can exit (enter) from (in) the bubble before the bubble reaches its end (after the bubble already starts) and you can see a counterexample in Fig. 2b. Finally, we require that all the pre- and post-events of a bubble must be part of the same sequential component; a counterexample is shown in Fig. 3.

The formal definition is stated as follow.

**Definition 1.** Let  $N_i = (B_i, E_i, F_i, m_0^i)$  be a SMD-EN System, for  $i = 1, 2$ . An  $\alpha$ -morphism from  $N_1$  to  $N_2$  is a total surjective map  $\varphi : X_1 \rightarrow X_2$  such that:

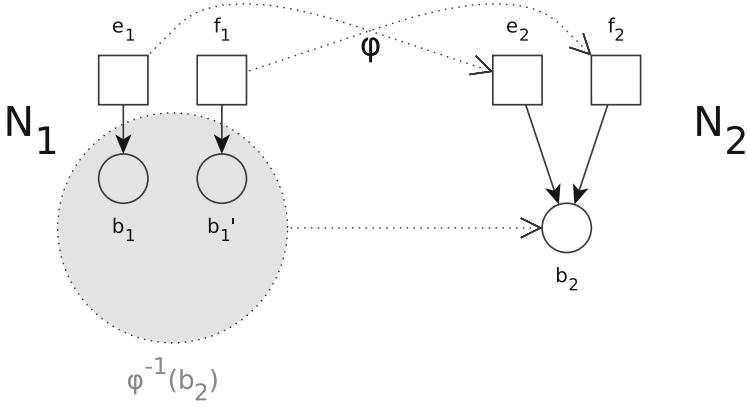
1.  $\varphi(B_1) = B_2$ ;
2.  $\varphi(m_0^1) = m_0^2$ ;
3.  $\forall e_1 \in E_1$ , if  $\varphi(e_1) \in E_2$ , then  $\varphi(\bullet e_1) = \bullet \varphi(e_1)$  and  $\varphi(e_1 \bullet) = \varphi(e_1) \bullet$ ;
4.  $\forall e_1 \in E_1$ , if  $\varphi(e_1) \in B_2$ , then  $\varphi(\bullet e_1 \bullet) = \{\varphi(e_1)\}$ ;
5.  $\forall b_2 \in B_2$ 
  - (a)  $N_1(\varphi^{-1}(b_2))$  is an acyclic net;
  - (b)  $\forall b_1 \in {}^\circ N_1(\varphi^{-1}(b_2))$ ,  $\varphi(\bullet b_1) \subseteq \bullet b_2$  and  $(\bullet b_2 \neq \emptyset \Rightarrow \bullet b_1 \neq \emptyset)$ ;
  - (c)  $\forall b_1 \in N_1(\varphi^{-1}(b_2))^\circ$ ,  $\varphi(b_1 \bullet) = b_2 \bullet$ ;
  - (d)  $\forall b_1 \in \varphi^{-1}(b_2) \cap B_1$ ,  
 $(b_1 \notin {}^\circ N_1(\varphi^{-1}(b_2)) \Rightarrow \varphi(\bullet b_1) = \{b_2\})$  and  $(b_1 \notin N_1(\varphi^{-1}(b_2))^\circ \Rightarrow \varphi(b_1 \bullet) = \{b_2\})$ ;
  - (e)  $\forall b_1 \in \varphi^{-1}(b_2) \cap B_1$ , there is a sequential component  $N_{SC}$  of  $N_1$  such that  $b_1 \in B_{SC}$  and  $\varphi^{-1}(\bullet b_2 \bullet) \subseteq E_{SC}$ .

We require that the map is total and surjective because  $N_1$  refines the abstract model  $N_2$ , and any abstract element must be related to its refinement.

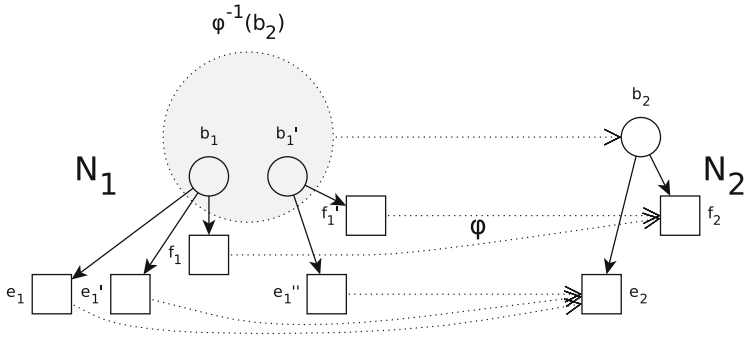
In particular, a subset of nodes can be mapped on a single condition  $b_2 \in B_2$ ; in this case, we will call *bubble* the subnet identified by this subset, and denote it by  $N_1(\varphi^{-1}(b_2))$ ; if more than one element is mapped on  $b_2$ , we will say that  $b_2$  is *properly refined* by  $\varphi$ .

As we show also in Figs. 4 and 5, in-conditions and out-conditions have different constraints, 5b and 5c respectively. As required by 5c, choices which are internal to a bubble cannot constrain a final marking of that bubble: i.e., each out-condition of the bubble must have the same choices of the condition it refines. Instead, pre-events do not need this strict constraint (5b): hence it is sufficient that pre-events of any in-condition are mapped on a subset of the pre-events of the condition it refines. Moreover, the conditions that are internal to a bubble must have pre-events and post-events which are all mapped to the refined condition  $b_2$ , as required by 5d. For example, in this particular case, we know that the choice between  $e_1$  and  $f_1$  of Fig. 4 is made before the bubble, and this is implied also by the requirement 5e on sequential components. Moreover, the conditions that are internal to a bubble must have pre-events and post-events which are all mapped to the refined condition  $b_2$ , as required by 5d, see also Fig. 6.

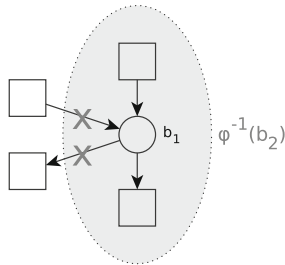
By requirement 5e, events in the neighbourhood of a bubble are not concurrent, and the same holds for their images. Within a bubble, there can be concurrent events; however, post-events are in conflict, and firing one of them



**Fig. 4.** Pre-events of an in-condition



**Fig. 5.** Post-events of an out-condition



**Fig. 6.** Constraints on an internal condition



will empty the bubble [6]. Moreover, given that a bubble can be abstracted by a single condition no input event of a bubble is enabled whenever a token is within the bubble [6].

By the combined effect of 5a–5e, in any execution, when a post-event of a bubble fires, in the next marking no local state within the bubble will be marked.

Note that cycles outside the bubbles are preserved and reflected by the morphism: this is given by the finiteness of a Petri Net and by the constraints on the environment of a node.

It is possible to show that the family of SMD-EN Systems together with  $\alpha$ -morphisms forms a category [6].

In [1, 6] structural and behavioral properties preserved or reflected by  $\alpha$ -morphisms has been studied. In particular, sequential components are reflected in the sense that the inverse image of a sequential component is covered by sequential components and  $\alpha$ -morphisms preserve reachable markings.

Moreover, stronger properties hold under additional constraints. In order to present them, we have to consider the following construction. Given an  $\alpha$ -morphism  $\varphi : N_1 \rightarrow N_2$ , and a condition  $b_2 \in B_2$  with its refinement, we define two new auxiliary SMD-EN Systems. The first one, denoted  $S_1(b_2)$ , contains the following elements: a copy of the subnet which is the refinement of  $b_2$ , i.e.: the bubble; its pre- and post-events in  $E_1$  and two new conditions, denoted  $b_1^{in}$  and  $b_1^{out}$ .  $b_1^{in}$  is pre-condition of all the pre-events, and  $b_1^{out}$  is post-condition of all the post-events. The initial marking of  $S_1(b_2)$  will be  $\{b_1^{in}\}$  or, if there are no pre-events, the initial marking of the bubble in  $N_1$ . The second system, denoted  $S_2(b_2)$ , contains  $b_2$ , its pre- and post-events and two new conditions:  $b_2^{in}$ , which is pre-condition of all the pre-events, and  $b_2^{out}$ , which is post-condition of all the post-events. The initial marking of  $S_2(b_2)$  will be  $\{b_2^{in}\}$  or, if there are no pre-events, the initial marking of  $b_2$ . Define  $\varphi^S$  as a map from  $S_1(b_2)$  to  $S_2(b_2)$ , which restricts  $\varphi$  to the elements of  $S_1(b_2)$ , and extends it with  $\varphi^S(b_1^{in}) = b_2^{in}$  and  $\varphi^S(b_1^{out}) = b_2^{out}$ . Note that  $S_1(b_2)$  and  $S_2(b_2)$  are SMD-EN Systems and that  $\varphi^S$  is an  $\alpha$ -morphism. Let  $Unf(S_1(b_2))$  be the unfolding of  $S_1(b_2)$ , with folding function  $u : Unf(S_1(b_2)) \rightarrow S_1(b_2)$ .

Consider the following additional constraints:

- c1** the initial marking of each bubble is at the start of the bubble itself; formally, for each  $b_2 \in B_2$  one of the following conditions hold:
  - $\varphi^{-1}(b_2) \cap m_0^1 = \emptyset$  or
  - if  $\bullet b_2 \neq \emptyset$  then there is  $e_1 \in \varphi^{-1}(\bullet b_2)$  such that  $\varphi^{-1}(b_2) \cap m_0^1 = e_1 \bullet$  or
  - if  $\bullet b_2 = \emptyset$  then  $\varphi^{-1}(b_2) \cap m_0^1 = {}^\circ \varphi^{-1}(b_2)$ ;
- c2** any condition is refined by a subnet such that, when a final marking is reached, this one enables events which correspond to the post-events of the refined condition, i.e.:  $\varphi^S \circ u$  is an  $\alpha$ -morphism from  $Unf(S_1(b_2))$  (in which we put a token in the in-condition of the net) to  $S_2(b_2)$ ;
- c3** different bubbles do not “interfere” with each other, where we say that two bubbles interfere with each other when their images share at least a neighbour.

Note that the third constraint is not restrictive since the refinement of two interfering conditions can be done in two different steps.

Under **c1**, **c2**, and **c3**, the following properties can be proved [6]:

**p1** reachable markings of  $N_2$  are reflected:

for all  $m_2 \in [m_0^2]$ , there is  $m_1 \in [m_0^1]$  such that  $\varphi(m_1) = m_2$ ;

**p2**  $N_1$  and  $N_2$  are weakly bisimilar:

by using  $\varphi$ , define two labelling functions,  $l_1$  and  $l_2$ , such that  $E_2$  are all observable, i.e.:  $l_2$  is the identity function, and  $l_1$  is defined as follows:  $\forall e \in E_1 : l_1(e) = \varphi(e)$  if  $\varphi(e) \in E_2$ ,  $l_1(e) = \tau$  if  $\varphi(e) \in B_2$ .

Then  $(N_1, l_1)$  and  $(N_2, l_2)$  are weakly bisimilar  $(N_1, l_1) \approx (N_2, l_2)$ .

As a result of property **p1**, modeling a system by refinement has an important advantage in model checking. Since reachable markings of  $N_2$  is reflected by  $\varphi$ , model checking for some properties can be done on  $N_2$  instead of  $N_1$ . Having an  $\alpha$ -morphism from  $N_1$  to  $N_2$ , if the additional constraints defined above hold, we get weak bisimulation between  $N_1$  and  $N_2$  as stated in property **p2**. One of the results of weak bisimulation is that the two nets preserve some properties like deadlock-freeness. Thus, one can deduce that the refined net  $N_1$  is deadlock-free iff  $N_2$  is deadlock-free.

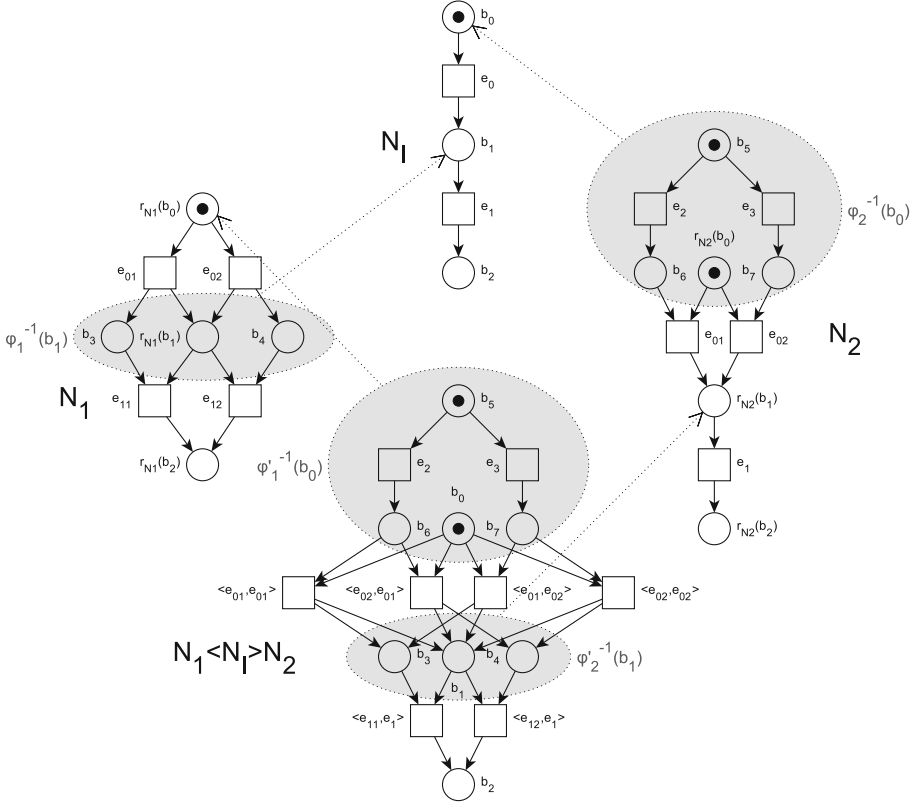
## 4 Composition Based on $\alpha$ -morphisms

In this section, we recall the composition of SMD-EN Systems based on  $\alpha$ -morphisms as defined in [1], on the basis of an example given in Fig. 7.

The two systems to be composed must be mapped onto a common *interface*, which is another SMD-EN System. The interface can be seen, intuitively, as a protocol of interaction, with which the components must comply, or as a common abstraction; in this second view, each component can refine some parts of the common abstraction. The two  $\alpha$ -morphisms, from the components to the interface, determine how the two components refine the local states of the interface, and then which elements are to be identified and which events in the two components have to synchronize.

In order to be composed, two net systems must be *canonical* with respect to the corresponding morphisms towards the interface. We say that a net system is canonical with respect to an  $\alpha$ -morphism if each bubble contains a condition, called *representation*, that corresponds to the abstraction of that bubble. Examples of representations are  $r_{N_1}(b_1)$  and  $r_{N_2}(b_0)$  in Fig. 7. If a system is not canonical, it is always possible to construct its unique (up to isomorphism) canonical version by adding the missing representations, and marking them as their images, or by deleting the multiple ones. Because of the constraints on  $\alpha$ -morphisms, and in particular of the ones on sequential components, point 5e of Definition 1, this construction does not modify the behavior of the original system and the corresponding modified morphism is still an  $\alpha$ -morphism.

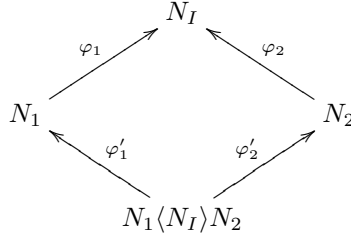
In the example given in Fig. 7 the interface  $N_I$  is a simple sequence of two events. The two components,  $N_1$  and  $N_2$ , refine two different local states,  $b_1$  and  $b_0$ , each one by a subnet, shown on a gray background.



**Fig. 7.** An example of composition based on  $\alpha$ -morphisms

The composed net  $N = N_1 \langle N_I \rangle N_2$  contains the refinement of each condition of the interface as it is in the two components, but for the representation, plus the condition itself, as we can see for condition  $b_0$  and  $b_1$  of the example. The rest of the net, not refined by the components, is taken as it is, but for the synchronizations of the events in the neighborhood of the refinements/bubbles. Such events must be synchronized so that each possible pair composed by one event of a component and one event of the other component must be created, as we can see for events mapped on events  $e_0$  and  $e_1$  of Fig. 7. Then, also arcs between in- and out-conditions of each bubble and its synchronized pre- and post-events must be created accordingly to the components. The initial marking is the union of the ones in the components. By construction,  $N = N_1 \langle N_I \rangle N_2$  is an EN System, and it is covered by sequential components [6].

This construction leads to the definition of a map  $\varphi'_i$  from  $N = N_1 \langle N_I \rangle N_2$  onto  $N_i$ ,  $i = 1, 2$ , relating each element local to a component to the corresponding representation and projecting synchronized events. In [6] it is proved that this map is an  $\alpha$ -morphism and that the following diagram commutes.



These results say that the composed system refines both the components, as well as the interface. The main result relating behavioral properties of the composed system to behavioral properties of its components is stated in the following proposition [1,6].

**Proposition 1.** *Let  $N_i = (B_i, E_i, F_i, m_0^i)$  be an SMD-EN System for  $i = 1, 2, I$ . Let  $\varphi_i$ , with  $i = 1, 2$ , be an  $\alpha$ -morphism from  $N_i$  to  $N_I$ , and let  $N = N_1 \langle N_I \rangle N_2$  be the composition of  $N_1$  and  $N_2$  using  $\varphi_1$  and  $\varphi_2$ . If  $N_1$  is weakly bisimilar to  $N_I$  then  $N = N_1 \langle N_I \rangle N_2$  is weakly bisimilar to  $N_2$ .*

Where, the labelling functions are derived from  $\varphi_1$  and  $\varphi'_2$ , respectively, in such a way that  $E_I$  and  $E_2$  are all observable and the invisible events of  $E_1$  and  $E$  are the ones which are mapped to conditions by  $\varphi_1$  and  $\varphi'_2$ , respectively.

This result tells us, in particular, that the composition of refinements  $N_1$  and  $N_2$ , which are weakly bisimilar to a common interface  $N_I$ , yields a system  $N$  which is weakly bisimilar to  $N_I$ ; and then, since bisimulation preserves deadlock-freeness, it is possible to deduce that  $N$  is also deadlock-free by verifying that  $N_I$  is deadlock-free. Remember that by **p2** it is possible to check weak bisimilarity between two systems related by an  $\alpha$ -morphism by considering their behavior only locally, as required by **c1**, **c2**, and **c3**.

## 5 Distributed Private Key Generation for id-Based Cryptography

In an id-based cryptographic system, unlike in the other public key cryptographic systems, a publicly known string such as e-mail address, domain name, a physical IP address or a combination of more than one string is used as public key. The idea of id-based cryptography was first proposed by Shamir in [12]. The proposed scheme enables users to communicate securely and to verify signatures without exchanging any private or public key. Consequently, there is no need for a certification authority to verify the association between public keys and users.

Basically, in an id-based cryptographic system there is a private key generator (PKG) which generates private keys for users. A PKG has a key pair which is referred to as *master key pair* consisting of a master private key and a master public key. A PKG generates a private key for a user by first hashing the user's

publicly known unique identity string, then signing the hashed id by the master private key. Later, the user can verify its key by using the master public key.

Since the PKG can generate private keys for users, it can sign or decrypt a message for any user or it can make users' private keys public. This is called the key escrow problem. Distributed private key generation (DPKG) is one of the effective solutions to the key escrow problem. In the schemes presented in [3, 4], secret sharing methods are used for distributing private key generation among multiple PKGs.

In a DPKG, a number of PKG nodes equally share the responsibility. In our work we followed the identity based distributed private key generation schemes presented in [3, 4]. For more details about the algorithms and the terminology it is recommended to refer to these citations.

The components of a DPKG system are divided into two main groups as PKG nodes and clients. PKG nodes are responsible for generating private keys for clients in a distributed manner. There is also an auxiliary component called bulletin board which is responsible for managing the global system variables, collecting the commitments from PKG nodes, calculating the final commitment and broadcasting all commitments.

We can examine the DPKG protocol in three steps: setup, distribution and extraction. Setup is a preparation step to create the system parameters and to get ready for extracting the master key pair distributively and for extracting the private keys. In this step, the bulletin board is given a security parameter and chooses some system variables according to this given security parameter; it then broadcasts public system parameters to be used by the other system components. It also initializes the commitment values to zero in order to set them to the values it will receive from PKG nodes in the distribution step. The final commitment is also set to zero; it will be calculated using the received commitments and will be broadcast later.

In the distribution step,  $n$  PKG nodes create a master private key together without using any dealer in such a way that the key cannot be reconstructed without retrieving  $k$  shares from these  $n$  PKGs.  $k$  is the threshold number of PKG nodes needed to collaborate together in order to construct the key. To do this, an improved version of  $(n, k)$  Feldman's secret sharing scheme stated in [4] is used. The idea behind secret sharing without a dealer is to make each PKG node create a secret of their own and calculate subshares to distribute among other PKG nodes. At the end, each PKG node will have  $n$  subshares including the one it calculated for itself. The sum of these subshares will be the share of the PKG node for the master private key. During the calculation of the subshares each PKG node also creates commitments corresponding to the subshares calculated by them. These commitments are sent to the bulletin board to be used by the PKG nodes for the verification of the received subshares. Note that in this DPKG system none of the PKG nodes knows the master secret key since each of them has only a part of it.

In the extraction step, a client with identity string  $ID$  contacts  $k$  available nodes from the PKG nodes pool. Each  $PKG_i$  signs the hashed identity string of

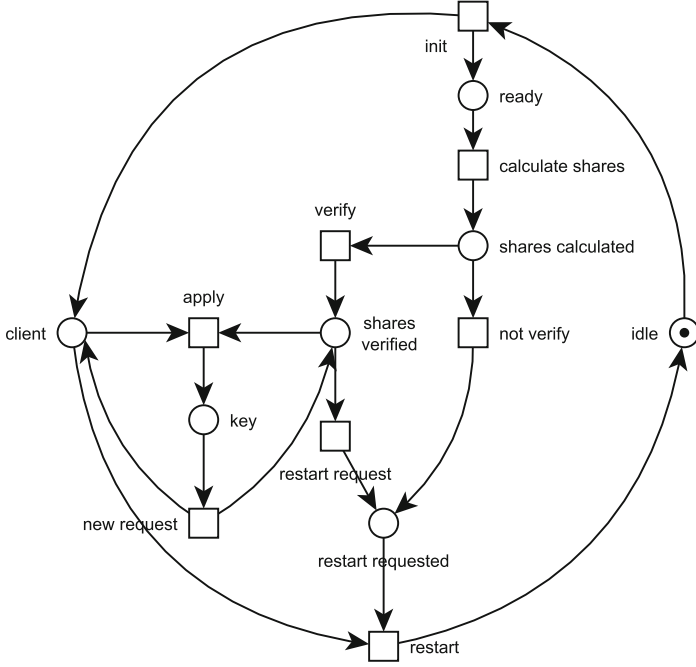
the client with its master private key share and returns a private key piece as  $s_i H(ID)$  over a secure and authenticated channel. After receiving  $k$  pieces from  $k$  available PKG nodes, the client constructs its private key. The client can verify the key by using bilinear pairings as stated in [4, 5].

## 6 The Model of DPKG

In this section, we present a Petri net model of a simplified DPKG system with three PKG nodes while the threshold number is two. We fixed these numbers for the sake of simplicity but the generated model is more generic and can easily be modified for different threshold and PKG node number as it will be discussed through this section. Our model consists of the following three nets:  $N_I$ ,  $N_{PKG}$  and  $N_C$ .  $N_I$  is the common interface between  $N_{PKG}$  and  $N_C$ . It is an abstract model of the whole system which represents the interaction between the main components of the system. This model also includes the abstract behavior of the bulletin board which is basically responsible for managing the global system variables and commitments.  $N_{PKG}$  is the net representing the behavior of PKG nodes while  $N_C$  is the net representing the behavior of clients in the DPKG system. We aim to compose  $N_{PKG}$  and  $N_C$  using  $N_I$  as the common interface and prove that the composed net  $N_{PKG} \langle N_I \rangle N_C$  and the interface  $N_I$  preserve and reflect some properties presented in Sect. 3 since there is an  $\alpha$ -morphism both from  $N_{PKG}$  to  $N_I$  and from  $N_C$  to  $N_I$ .

The net  $N_I$ , which represents the interface, is given in Fig. 8. This net is an abstract model of the behavior of all three system components: bulletin board, PKG nodes and clients. The system is idle in the beginning. After event *init* occurs, system components are initialized and all PKG nodes are ready for generating a secret key in a distributed way. The event *init* includes the setup step of the protocol which is explained in Sect. 5. The condition *calculate shares* represents the whole process including calculating subshares and exchanging between PKG nodes in order to calculate their shares for the master private key. During this, each PKG node chooses a secret polynomial. It calculates the commitment corresponding to its secret polynomial and sends it to the bulletin board. It also calculates  $n$  subshares using its polynomial where  $n$  is the number of PKG nodes in the system. Each PKG node sends the subshare to the related PKG node. After exchanging is completed, each PKG node will have  $n - 1$  subshares sent by other PKG nodes and one subshare of its own. By using these  $n$  subshares each PKG node calculates its share. When the condition *shares calculated* becomes true, it means that all the PKG nodes finished calculating share and each of them is holding a share.

Once PKG nodes have their shares, they can verify their shares using the final commitment value which is already calculated during the abstract event *calculate shares*. If all the shares are correctly verified, the condition *shares verified* becomes true so a client can apply for extracting a private key. In this model, event *apply* includes choosing  $k$  available PKG nodes, receiving  $k$  pieces and calculating its private key using these pieces. When the condition *key* is



**Fig. 8.**  $N_I$ , the net representing the interface.

true, the client has a key but we do not know if the key is correct or not by looking in this abstract model. In both cases *new request* event can occur then the system can continue with a restart which repeats the whole distributed private key generation. In case of a fail during the verification of shares, the system is forced to a restart without a key extraction.

$N_I$  is a live and reversible net which means that from any reachable marking one can always get back to the initial state. These two properties are important because a DPKG system must always be alive to respond to the clients' key requests and key generation process must be restartable whenever it is needed. The net  $N_I$  is also covered by sequential components which is a requirement in order to be able to look for an  $\alpha$ -morphism. The sequential components covering the net can be shown as lists of conditions:  $\{\text{idle}, \text{ready}, \text{shares calculated}, \text{shares verified}, \text{key}, \text{restart requested}\}$ ,  $\{\text{idle}, \text{client}, \text{key}\}$ .

Figure 9 shows the net  $N_{PKG}$ . This net refines the interface with respect to PKG nodes' behavior. All the elements of  $N_{PKG}$  are mapped to the element with the same name in  $N_I$  but for the subnet circled by dashed line that is mapped to a single condition. This subnet forms a bubble which is a refinement of the condition *shares calculated* in  $N_I$ . The bubble shows the calculation and exchange of subshares between three PKG nodes and calculation of shares by each PKG node whereas in  $N_I$  the occurrence of the whole process is abstracted

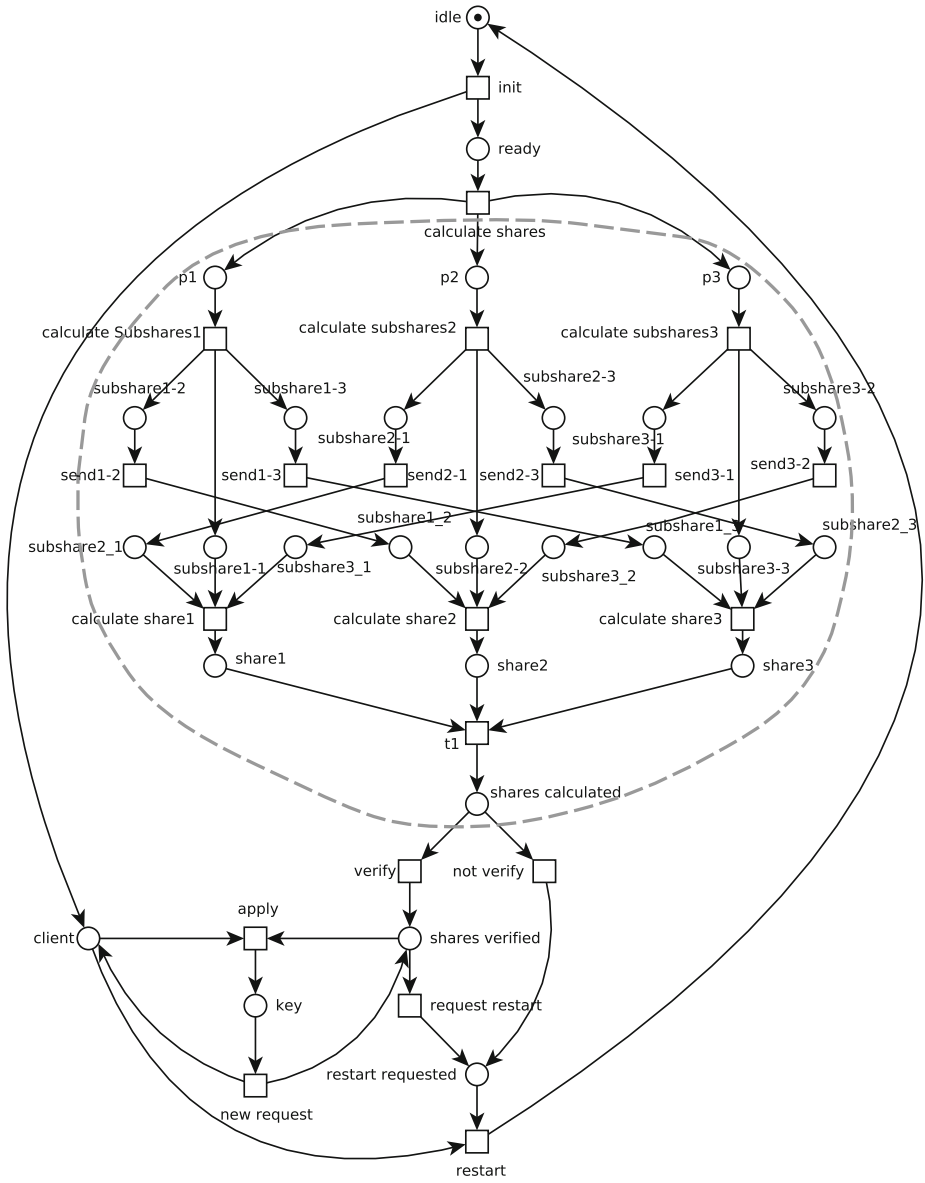


Fig. 9.  $N_{PKG}$ , the net representing the PKG nodes.

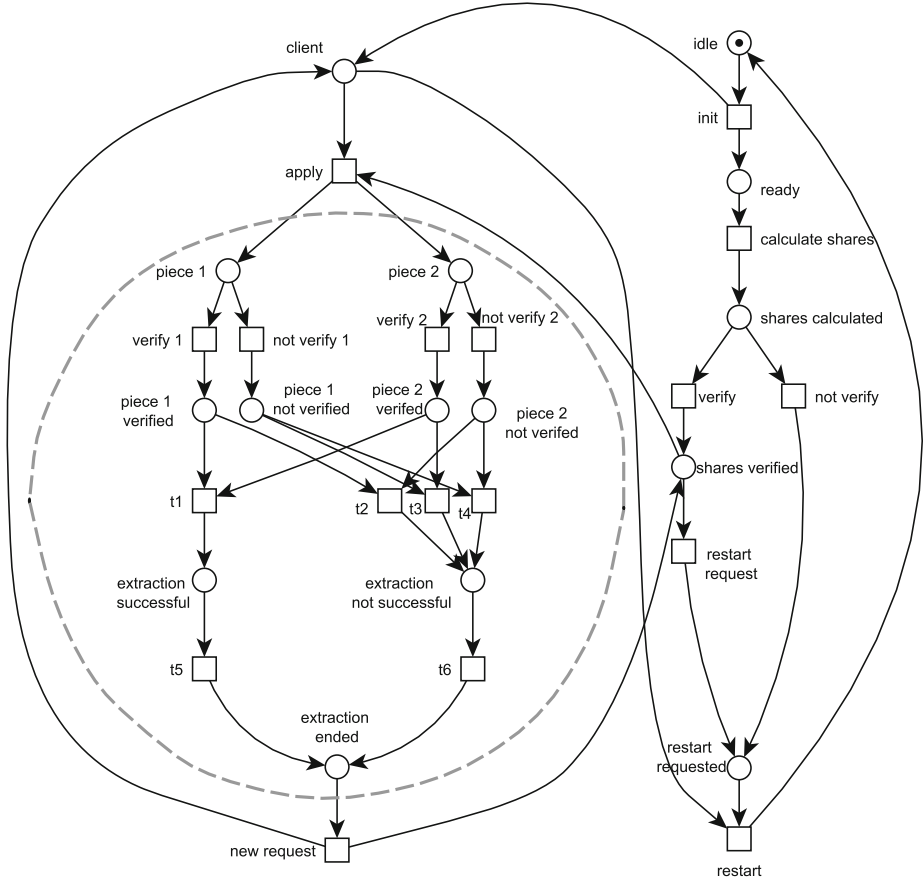


by one condition. If we model a system with  $n$  PKG nodes instead of three nodes, only the bubble will grow, the other elements of the net will remain the same.

$N_{PKG}$  is also live, reversible and covered by sequential components like  $N_I$ . It is already shown that the conditions outside the bubble are covered by sequential components. Thus, here we will only show how the bubble is covered by sequential components. After event *calculate shares* the net branches into three paths and after each event *calculate subshares*  $i$  for  $i = 1, 2, 3$  the net branches again into three paths. This fact results in having nine sequential components inside the bubble. Here we present only some of the sequential components as the lists of conditions that construct the components:  $\{p1, \text{subshare } 1-2, \text{subshare } 1-2, \text{share } 2, \text{shares calculated}\}$ ,  $\{p1, \text{subshare } 1-1, \text{share } 1, \text{shares calculated}\}$ ,  $\{p1, \text{subshare } 1-3, \text{subshare } 1-3, \text{share } 3, \text{shares calculated}\}$ . The paths starting with conditions  $p2$  and  $p3$  can also be constructed in the same way.

In order to prove that there is an  $\alpha$ -morphism from  $N_{PKG}$  to  $N_I$  we have to show that the requirements in Definition 1 are satisfied. To begin with, the initial states of  $N_{PKG}$  and  $N_I$  are related. For all the events in  $N_{PKG}$  which are mapped to an event in  $N_I$ , also the pre-conditions and post-conditions of these events are mapped to the pre- and post-conditions of the related events in  $N_I$ . Moreover, for all the events in  $N_{PKG}$  that are mapped to a condition in  $N_I$ , all the pre- and post-conditions of that event are also mapped to the same condition in  $N_I$ . We see that the nets satisfy the first four requirements of  $\alpha$ -morphism. To continue with, we can see that the bubble in  $N_{PKG}$  is acyclic so 5a is satisfied. As seen in Fig. 9 all the in-elements of the bubble are generated by the only one event entering the bubble which is mapped to the corresponding event in the interface, *calculate shares*. It is also seen that post-events of the out-condition of the bubble are exactly the same post-events of the corresponding condition in the interface. Thus, 5b and 5c are satisfied. 5d is also satisfied because the conditions that are internal to the bubble have pre-events and post-events which are all mapped to the refined condition *shares calculated* in  $N_I$  but for in- and out-elements. Finally, as we already listed the sequential components of the net, it is easy to see that for each condition of the bubble there is a sequential component containing that condition and all the pre- and post-events of the bubble, so requirement 5e is satisfied. In this way, we proved that there is an  $\alpha$ -morphism from  $N_{PKG}$  to  $N_I$ .

The net shown in Fig. 10,  $N_C$ , is the net representing the behavior of a client. While it includes the whole abstract model, it refines the key extraction process of a client. The bubble shown with a dashed line is the refinement of the condition *key* in the interface  $N_I$ . In this refinement, receiving two pieces from chosen PKG nodes, calculating the private key and verification of it, is modeled in more details. In a DPKG system where the threshold number is two, when a client applies for a private key, it receives two pieces from two available PKG nodes. The client can verify the pieces it received. If both pieces are verified then the client can extract its private key by using these pieces and the system reaches a state where extraction is successful. In case at least one of the pieces are not verified then the condition *extraction not successful* becomes true. After

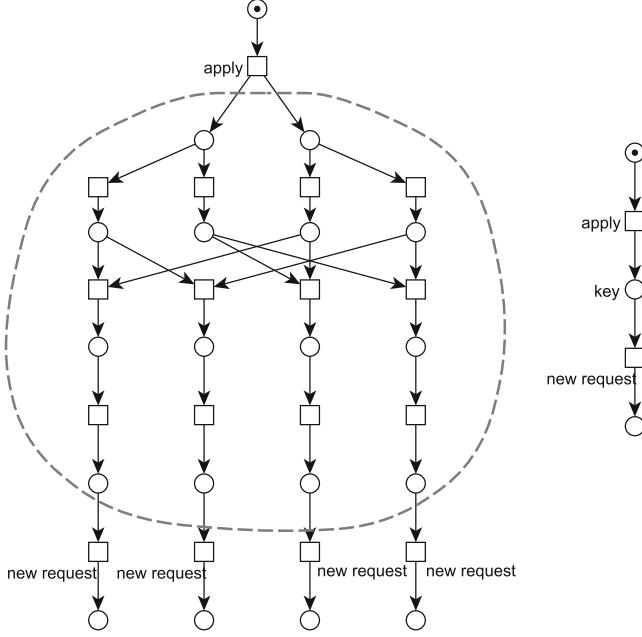


**Fig. 10.**  $N_C$ , the net representing the clients.

both failed or successful extraction, the system reaches a state where *extraction ended* is true and a new key can be requested by the same client or by any other client in the system. Again if we improve the model for threshold value  $k$  instead of two, only the bubble will grow but the other elements of the net will remain the same.

This net is also live, reversible and covered by sequential components. Here we give the sequential components which are enough to cover the net as lists of conditions:  $\{\text{idle}, \text{ready}, \text{shares calculated}, \text{shares verified}, \text{piece 1}, \text{piece 1 verified}, \text{piece 1 not verified}, \text{extraction successful}, \text{extraction not successful}, \text{extraction ended}, \text{restart requested}\}$ ,  $\{\text{idle}, \text{ready}, \text{shares calculated}, \text{shares verified}, \text{piece 2}, \text{piece 2 verified}, \text{piece 2 not verified}, \text{extraction successful}, \text{extraction not successful}, \text{extraction ended}, \text{restart requested}\}$ ,  $\{\text{client}, \text{idle}, \text{piece 1}, \text{piece 1 verified}, \text{piece 1 not verified}, \text{extraction successful}, \text{extraction not successful}, \text{extraction ended}\}$ .

It is very easy to see that the first four requirements of  $\alpha$ -morphism are already satisfied so we can continue with checking the rest of the requirements.

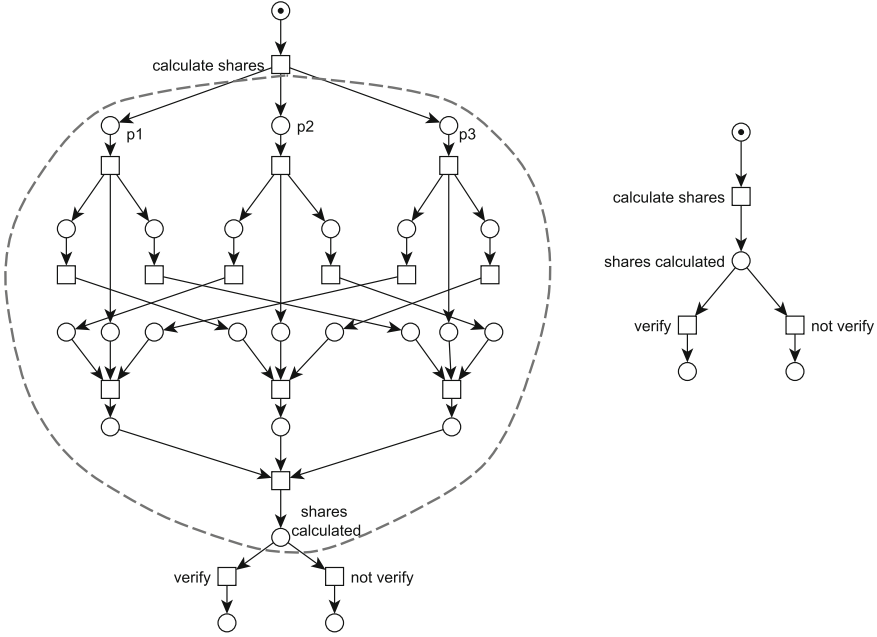


**Fig. 11.**  $Unf(S_C(key))$  and  $S_I(key)$

The bubble contains no cycles so 5a of Definition 1 is satisfied. All the in-elements of the bubble are generated by the only one event entering the bubble which is mapped to the corresponding event in the interface, *apply*. There is also only one post-event of out-condition of the bubble which empties the bubble and this event is mapped to the post-event of *key*. With these observation it is easy to see that (with reference to Definition 1) 5b and 5c are satisfied. 5d is also satisfied because the conditions that are internal to the bubble have pre-events and post-events which are all mapped to the refined condition *key* in  $N_I$  but for in- and out-elements.

Finally, as we already listed the sequential components of the net, it is easy to see that for each condition of the bubble there is a sequential component containing that condition and all the pre- and post-events of the bubble, so requirement 5e of Definition 1 is satisfied. Considering all the requirements, we can say that there is an  $\alpha$ -morphism between  $N_C$  and  $N_I$ .

Now that we proved that there is an  $\alpha$ -morphism both from  $N_{PKG}$  to  $N_I$  and from  $N_C$  and  $N_I$ , we can prove that the composed net is weakly bisimilar to the interface by showing that the additional requirements which are stated as **c1**, **c2**, and **c3** in Sect. 3 are satisfied by  $N_{PKG}$  and  $N_C$ . Proposition 1 states that if both of the components are weakly bisimilar to the interface, then the composed net is also weakly bisimilar to the interface. Thus, here we first show that  $N_C$  is weakly bisimilar to the interface  $N_I$ . To do this, we follow the construction of the two auxiliary nets given in Sect. 3, i.e., we consider the bubble in  $N_C$  and the



**Fig. 12.**  $Unf(S_{PKG}(\text{shares calculated}))$  and  $S_I(\text{shares calculated})$

corresponding condition  $key$  in  $N_I$  and we add their pre- and post-events to the subnets. We also add two more conditions to each subnet: one condition to be a pre-condition to all pre-events and another condition to be a post-condition to all post-events. Let us name these two subnets as  $S_C(key)$  and  $S_I(key)$ . Finally, we build the unfolding of  $S_C(key)$ , represented as  $Unf(S_C(key))$ . The resulting nets are shown in Fig. 11.

We follow the same procedure for  $N_{PKG}$  and we get two subnets  $Unf(S_{PKG}(\text{shares calculated}))$  and  $S_I(\text{shares calculated})$  as in Fig. 12. When we examine these subnets, we see that no condition of the bubbles is in the initial marking. Any condition is refined by a subnet such that, when a final marking is reached, this one enables events which correspond to the post-events of the refined condition, and there is an  $\alpha$ -morphism both from  $Unf(S_C(key))$  to  $S_I(key)$  and from  $Unf(S_{PKG}(\text{shares calculated}))$  to  $S_I(\text{shares calculated})$ . Thus, **c1** and **c2** are satisfied. Since there is only one bubble in both  $N_{PKG}$  and  $N_C$ , **c3** is automatically satisfied. Consequently, we can say that the additional properties **p1** and **p2** hold. The validity of these properties allows us to conclude that  $N_{PKG}$  and  $N_C$  are weakly bisimilar to  $N_I$  and, by considering Proposition 1, also that the composed net  $N_{PKG}\langle N_I \rangle N_C$  is weakly bisimilar to  $N_I$ . This fact, together with the one that our nets satisfy the requirements of  $\alpha$ -morphisms and the other three additional constraints, allows us to infer important behavioural properties of both the components  $N_{PKG}$  and  $N_C$  and of the composed net  $N_{PKG}\langle N_I \rangle N_C$ . In particular, being  $N_I$  live and reversible, we can directly conclude that also

$N_{PKG}$ ,  $N_C$  and  $N_{PKG}\langle N_I \rangle N_C$  are live and reversible. Moreover, the property **p1** of reflecting reachable markings allows us to solve some reachability problems in the composed net by analyzing corresponding reachability problems in the interface which is advantageous with respect to computational cost. To give an example, we can consider the existence of the following situation in the composed net  $N_{PKG}\langle N_I \rangle N_C$ : the condition *shares verified* should not be true while there is at least one token in a bubble. Performing a reachability analysis on the composed net is complex in terms of time and space since both the net and the logic formula we have to use to represent the global state to be investigated are big. Instead, the mentioned global state can be easily translated into a global state of the interface,  $N_I$ . Since each bubble in the composed net is mapped to a condition in the interface, reachability analysis becomes easier. The previously mentioned critical situation is reflected in the interface as the following: the condition *shares verified* cannot be true while *key* or *shares calculated* is true. Performing a reachability analysis for existence of this situation in  $N_I$  is easier than analyzing the composed net. Moreover, we do not even need to build the composed net.

## 7 Conclusion

We have developed a Petri net model of a protocol for distributed generation of private keys. The model has been obtained by composing two net models on a common interface. The first component models the interactions among PKG nodes, while the second component models clients of the key generator. Both components refine a common interface, representing the interactions among components.

We have then discussed behavioral properties of the model, directly derivable from properties of the components without generating the composed net. In particular, we have shown that some markings are not reachable.

On one hand, we have verified modeling and analysis capacity of the compositional approach proposed in [1] by means of a real world example. We have showed some advantages of  $\alpha$ -morphisms in modeling and model checking. On the other side, we have proposed a model of distributed private key generation protocol by using the compositional approach.

We now plan to explore how to extend the approach to other classes of Petri nets, particularly P/T nets and high-level nets. With respect to the model, we plan to improve it giving a less abstract specification in order to propose a formal verification of the protocol and to discuss its weak and strong aspects.

**Acknowledgements.** This work was partially supported by MIUR and by MIUR - PRIN 2010/2011 grant ‘Automi e Linguaggi Formali: Aspetti Matematici e Applicativi’, code H41J12000190001.

## References

1. Bernardinello, L., Mangioni, E., Pomello, L.: Local state refinement and composition of elementary net systems: an approach based on morphisms. In: Koutny, M., van der Aalst, W.M.P., Yakovlev, A. (eds.) *ToPNoC VIII*. LNCS, vol. 8100, pp. 48–70. Springer, Heidelberg (2013)
2. Bernardinello, L., Monticelli, E., Pomello, L.: On preserving structural and behavioural properties by composing net systems on interfaces. *Fundam. Inform.* **80**(1–3), 31–47 (2007)
3. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. *SIAM J. Comput.* **32**, 586–615 (2003)
4. Kate, A., Goldberg, I.: Asynchronous distributed private-key generators for identity-based cryptography. *IACR Cryptology ePrint Archive* 2009, 355 (2009)
5. Kilinc, G., Fovino, I.N., Ferigato, C., Koltuksuz, A.: A model of distributed key generation for industrial control systems. In: *11th Workshop on Discrete Event Systems*, Guadalajara, Mexico, vol. 11, pp. 356–363 (2012)
6. Mangioni, E.: Modularity for system modelling and analysis. Ph.D. thesis, Università degli Studi di Milano-Bicocca, Dottorato di ricerca in Informatica, ciclo 24 (2013)
7. Milner, R.: *Communication and Concurrency*. Prentice-Hall Inc., Upper Saddle River (1989)
8. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, part i. *Theor. Comput. Sci.* **13**, 85–108 (1981)
9. Nielsen, M., Rozenberg, G., Thiagarajan, P.S.: Elementary transition systems. *Theor. Comput. Sci.* **96**(1), 3–33 (1992)
10. Pomello, L., Rozenberg, G., Simone, C.: A survey of equivalence notions for net based systems. In: Rozenberg, G. (ed.) *APN 1992*. LNCS, vol. 609, pp. 410–472. Springer, Heidelberg (1992)
11. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) *APN 1998*. LNCS, vol. 1491, pp. 12–121. Springer, Heidelberg (1998)
12. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

Transactions on Petri Nets and Other Models of  
Concurrency IX

Koutny, M.; Haddad, S.; Yakovlev, A. (Eds.)

2014, XIII, 179 p. 84 illus., Softcover

ISBN: 978-3-662-45729-0