Modeling Distributed Private Key Generation by Composing Petri Nets

Luca Bernardinello[1], Görkem Kılınç[1], Elisabetta Mangioni[1,2], and Lucia Pomello[1]

[1] Dipartimento di Informatica Sistemistica e Comunicazione,
Università degli studi di Milano - Bicocca,
Viale Sarca, 336 - Edificio U14 - I-20126 Milano, Italia
[2] Istituto per la Dinamica dei Processi Ambientali,
Consiglio Nazionale delle Ricerche (CNR-IDPA),
Piazza della Scienza, 1 - Edificio U1 - I-20126 Milano, Italia

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 Systems (ENS) 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 ENS 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 [11], [4]. Later, the same operation was defined over a new class of morphisms, called α-morphisms (see [3] and [1]).

An α-morphism from an ENS N_1 to an ENS 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 Section 3.

When composing two ENS, 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 Section 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 analyse the composed system.

Ideally, one would like to derive behavioural 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 behavioural information about components and interface; however, this is limited to only a part of the models, and does not involve the whole system model.

Here, we test these ideas on a protocol for distributed generation of id-based cryptographic keys. The protocol, described in more detail in Section 5, requires the cooperation of several *private key generators* (PKGs) 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 [5]; an improved version is presented in [6]. In [7], a Petri net model for the protocol to be implemented on industrial control systems is presented.

In the next section, basic definitions related to ENS are recalled. Section 3 recalls the formal definition of α-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 ENS, based on α-morphisms, is informally recalled in Section 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 modelled by Petri nets in Section 6. In the same section, we analyze behavioural 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 [13].

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 ${}^\bullet x = \{y \in X | (y, x) \in F\}$; the *postset* of x is $x^\bullet = \{y \in X | (x, y) \in F\}$; the *neighbourhood* 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.

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 $N^\circ = \{x \in X : x^\bullet = \emptyset\}$.

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 $N(A) = (B \cap A, E \cap A, F \cap (A \times A))$. Given a subset of conditions $A \subseteq B$, we say that N_A is the *subnet of* N *generated* by A if $N_A = (A, {}^\bullet A^\bullet, F \cap ((A \cup {}^\bullet A^\bullet) \times (A \cup {}^\bullet A^\bullet)))$. Note that given $A \subseteq B$, $N(A \cup {}^\bullet A^\bullet) = N_A$.

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, 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 $m\,[e\rangle$, if ${}^\bullet e \subseteq m$ and $e^\bullet \cap m = \emptyset$; the occurrence of e at m leads from m to m', denoted $m\,[e\rangle\, m'$, iff $m' = (m \setminus {}^\bullet e) \cup e^\bullet$.

Let ϵ denote the empty word in E^*. The firing rule is extended to sequences of events by setting $m\,[\epsilon\rangle\, m$ and $\forall e \in E, \forall w \in E^*, m\,[ew\rangle\, m' = m\,[e\rangle\, m''[w\rangle m'$; 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 $m_0\,[w\rangle\, m$. The *set of all reachable markings* of N is denoted by $[m_0\rangle$.

An EN System is *contact-free* if $\forall e \in E, \forall m \in [m_0\rangle$: ${}^\bullet e \subseteq m$ implies $e^\bullet \cap m = \emptyset$. An EN System covered by Sequential Components is contact-free [13]. 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.

Let $N = (B, E, F)$ be a net, and let $x, y \in X$. We say that x and y are in *conflict*, denoted by $x \#_N y$, 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.

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.

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 $x \in X, \{y : yF^*x\}$ is finite; $\#_N$ is irreflexive and the minimal elements with respect to F^* are conditions. Occurrence nets were introduced in [10]; in [13] they are called branching process nets.

A branching process of 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 $\pi : P \cup T \to B \cup E$ be a map. The pair (Σ, π) is a *branching process* of N if $\pi(P) \subseteq B$, $\pi(T) \subseteq E$; π restricted to the minimal elements of Σ is a bijection on m_0; for each $t \in T$, π restricted to ${}^\bullet t$ is injective and π 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.*

3 α-morphisms

In this section we present the formal definition of α-morphisms [3, 2] for State Machine Decomposable Elementary Net Systems (SMD-EN Systems) and the structural and behavioural properties α-morphisms preserve and reflect.

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

In-conditions and out-conditions have different constraints, 5b and 5c respectively. As required by 5c, choices which are internal to a bubble can not 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. 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 will empty the bubble [8]. 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 [8].

It is possible to show that the family of SMD-EN Systems together with α-morphisms forms a category [8].

In [8] and [3] structural and behavioral properties preserved or reflected by α-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 α-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 α-morphism $\varphi : N_1 \to N_2$, 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 of all the pre-events, and b_1^{out} is post 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 of all the pre-events, and b_2^{out}, which is post 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 φ^S as a map from $S_1(b_2)$ to $S_2(b_2)$, which restricts φ 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 φ^S is an α-morphism. Let $Unf(S_1(b_2))$ be the unfolding of $S_1(b_2)$, with folding function $u : Unf(S_1(b_2)) \to S_2(b_2)$.

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 α-morphism from $\mathit{Unf}(S_1(b_2))$ (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 [8]:

p1 reachable markings of N_2 are reflected:
for all $m_2 \in [m_0^2\rangle$, there is $m_1 \in [m_0^1\rangle$ such that $\varphi(m_1) = m_2$;

p2 N_1 and N_2 are weakly bisimilar:
by using φ, define two labelling functions such that E_2 are all observable, i.e.: l_2 is the identity function, and the invisible events of N_1 are the ones mapped to conditions; then (N_1, l_1) and (N_2, l_2) are weakly bisimilar $(N_1, l_1) \approx (N_2, l_2)$.

For a definition of weak bisimulation of EN Systems see [12] and [9].

4 Composition based on α-morphisms

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

The two systems to be composed, N_1 and N_2, must be mapped onto a common *interface*, which is another SMD-EN System N_I. 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 α-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.

To compose two net systems, each must be *canonical* with respect to the corresponding morphism towards the interface. We say that a net system is canonical with respect to an α-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. 1. 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 α-morphisms, and in particular of the ones on sequential components, point 5e of Def. 1, this construction does not modify the behaviour of the original system and the corresponding modified morphism is still an α-morphism.

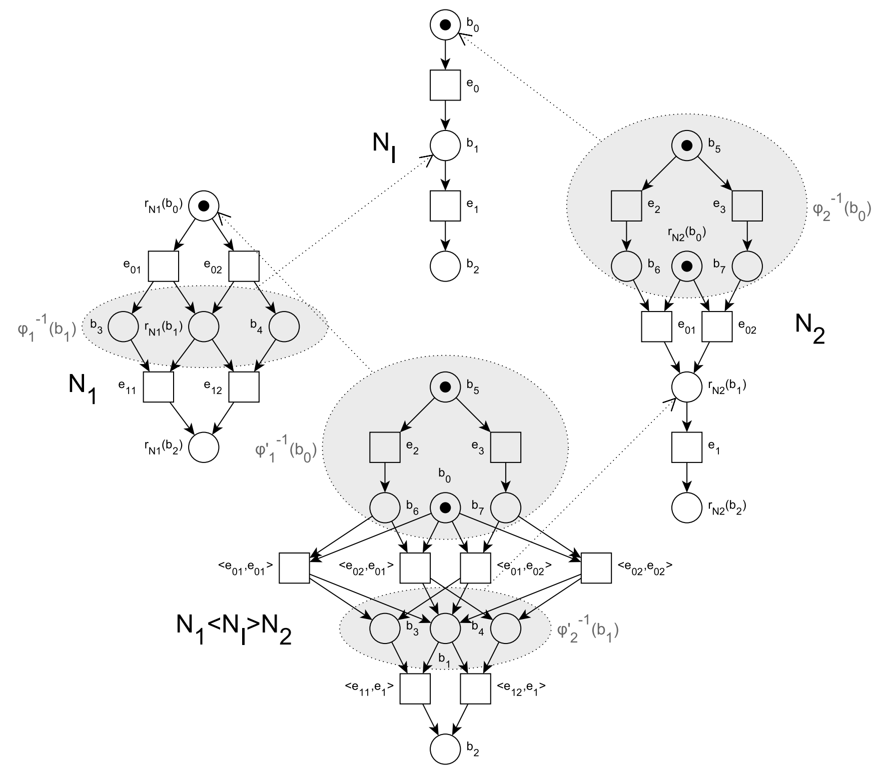

Fig. 1: An example of composition based on α-morphisms

In the example given in Figure 1 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.

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 neighbourhood 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. 1. Then, also arcs between in- and out-condition of each bubble and its pre and post (synchronized) 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 [8].

This construction leads to the definition of a map φ'_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 [8] it is proved that this map is an α-morphism and that the following diagram commutes.

$$\begin{tikzcd}
 & N_I & \\
N_1 \arrow[ur, "\varphi_1"] & & N_2 \arrow[ul, "\varphi_2"'] \\
 & N_1\langle N_I\rangle N_2 \arrow[ul, "\varphi'_1"] \arrow[ur, "\varphi'_2"'] &
\end{tikzcd}$$

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 [8], [2].

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 φ_i, with $i = 1, 2$, be an α-morphism from N_i to N_I, and let $N = N_1\langle N_I\rangle N_2$ be be the composition of N_1 and N_2 using φ_1 and φ_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 φ_1 and φ'_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 φ_1 and φ'_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 α-morphism by considering their behaviour 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 strings is used as public key. The idea of id-based cryptography was first proposed by Shamir in [14]. 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 as master key pair consisting of a master private key and a master public key. A PKG generates a private key for a user basically by first hashing its publicly known unique identity string then signing 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 problem about private key generation is called the key escrow problem. Distributed private key generation (DPKG) is one of the effective solutions to the key escrow problem. In both schemes [5], [6] secret sharing methods are used for distributing private key generation among multiple PKGs.

In a DPKG there is a number of PKG nodes participating while they share the responsibility equally. In our work we followed the identity based distributed private key generation schemes presented in [5] and [6]. 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 a third 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 these commitments.

We can examine 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 extraction of private keys. In this step, 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. Final commitment is also set to zero which will be calculated using the received commitments and it will be broadcast later.

Distribution step is illustrated in Figure 2. In this step, n PKG nodes create a master private key together without using any dealer in 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 [6] 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.

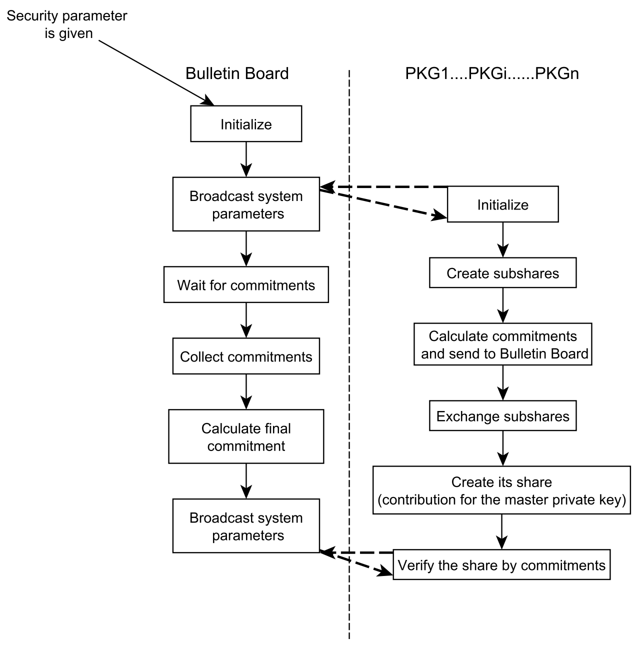
Security parameter is given
Bulletin Board
PKG1....PKGi......PKGn
Initialize
Broadcast system parameters
Wait for commitments
Collect commitments
Calculate final commitment
Broadcast system parameters
Initialize
Create subshares
Calculate commitments and send to Bulletin Board
Exchange subshares
Create its share (contribution for the master private key)
Verify the share by commitments

Fig. 2: Block schema of the distribution step of private key generation.

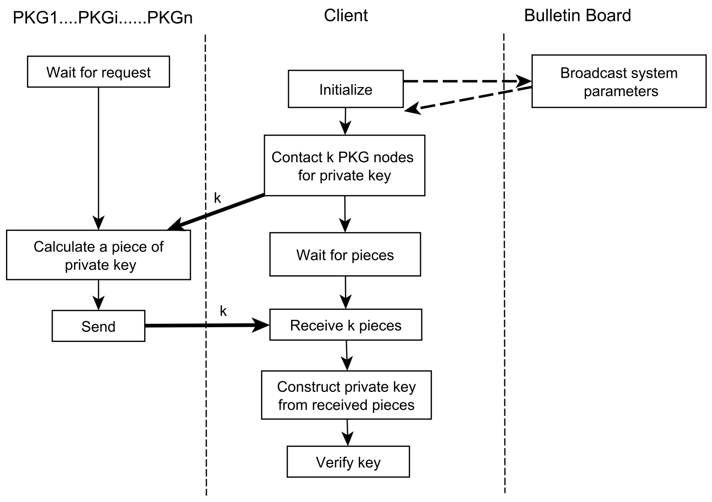
PKG1....PKGi......PKGn
Client
Bulletin Board
Wait for request
Initialize
Broadcast system parameters
Contact k PKG nodes for private key
k
Calculate a piece of private key
Wait for pieces
Send
k
Receive k pieces
Construct private key from received pieces
Verify key

Fig. 3: Block schema of the extraction step.

In extraction step, as it is illustrated in Figure 3, a client with identity string ID contacts k available nodes from the PKG nodes pool. Each PKG$_i$ signs hashed identity string of the client with its master private key share and returns a private key piece as $s_i\mathrm{H}(ID)$ over a secure and authenticated channel. After receiving k pieces from k available PKG nodes, client constructs its private key. The client can verify the key by using bilinear pairings as it is stated in [6] and [7].

6 The model of DPKG

In this section, we present Petri net model of a simplified DPKG system with three PKG nodes while the threshold number is two. We fixed these numbers for the 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 Section 3 since there is an α-morphism both from N_{PKG} to N_I and from N_C to N_I.

N_I, which is the net representing the interface, is given in Figure 4. 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 distributedly. The event *init* includes the setup step of the protocol which is explained in Section 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 bulleting 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 ver-*

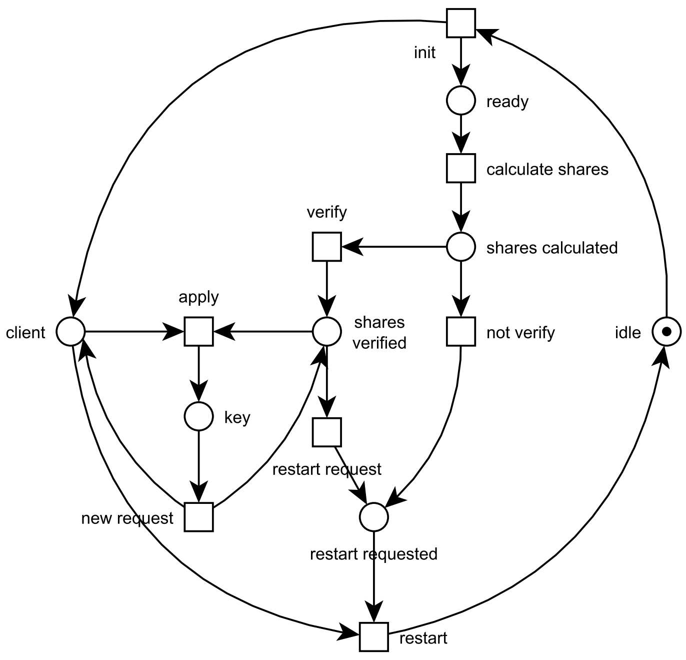

Fig. 4: N_I, the net representing the interface.

ified 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 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 or 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 α-morphism. The sequential components covering the net can be shown as lists of conditions: {*idle, ready, shares calculated, shares verified, key, restart requested* }, {*idle, client, key*}.

Figure 5 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

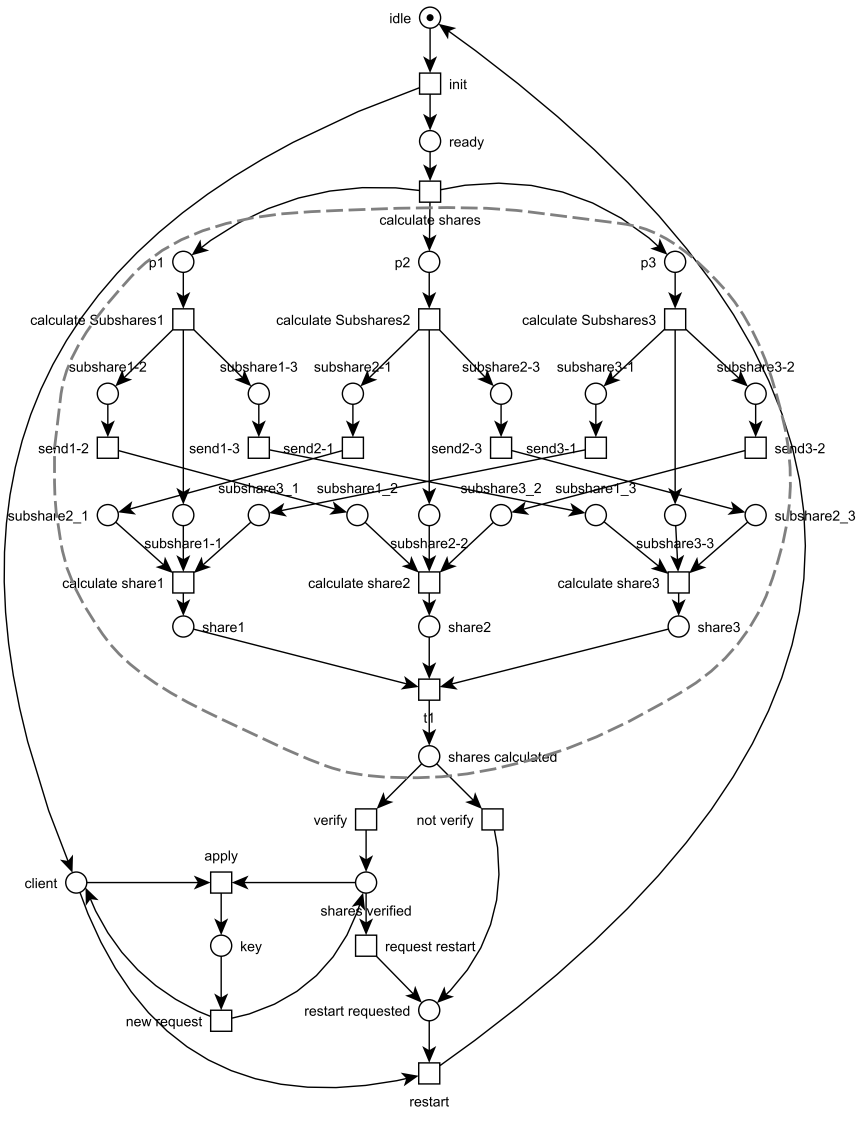

Fig. 5: N_{PKG}, the net representing the PKG nodes.

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 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, subshare 1-2, subshare 1_2, share 2, shares calculated*}, {*p1, subshare 1-1, share 1, shares calculated*}, {*p1, subshare 1-3, subshare 1_3, share 3, 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 α-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 α-morphism. To continue with, we can see that the bubble in N_{PKG} is acyclic so 5a is satisfied. As seen in Figure 5 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 α-morphism from N_{PKG} to N_I.

The net shown in Figure 6, 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

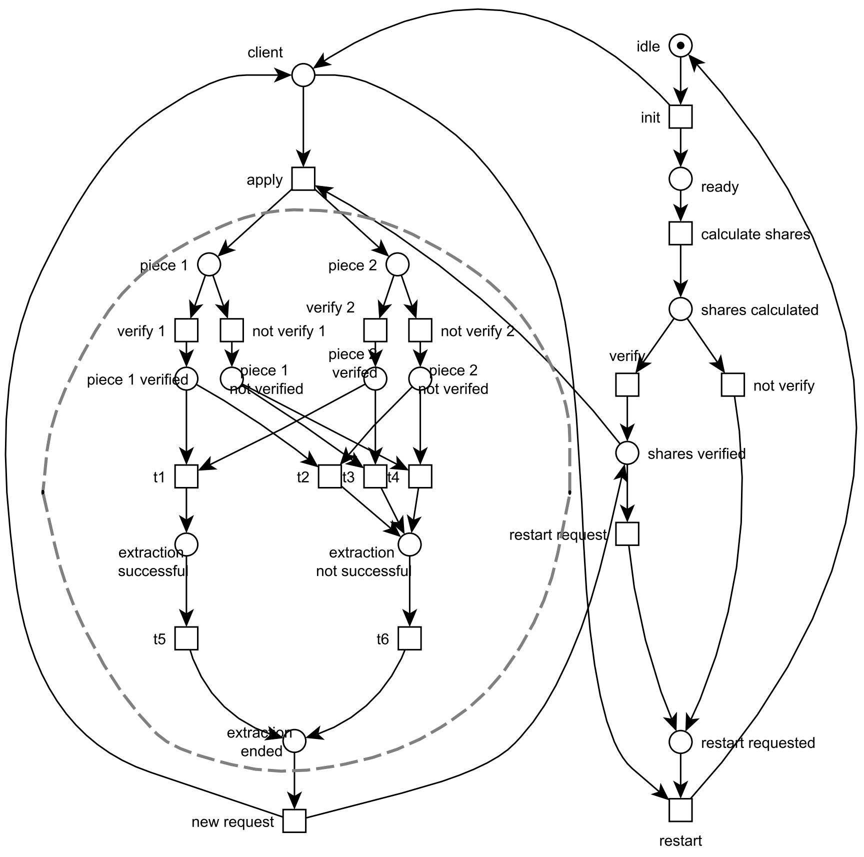

Fig. 6: N_C, the net representing the clients.

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 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: {*idle, ready, shares calculated, shares verified, piece 1, piece 1 verified, piece 1 not verified, extraction successful, extraction not successful, extraction ended, restart requested*}, {*idle, ready, shares calculated, shares verified, piece 2, piece 2 verified, piece 2 not verified, extraction successful, extraction not successful, extraction ended, restart requested*}, {*client, idle, piece 1, piece 1 verified, piece 1 not verified, extraction successful, extraction not successful, extraction ended*}.

It is very easy to see that the first four requirements of α-morphism are already satisfied so we can continue with checking the rest of the requirements. The bubble contains no cycles so 5a 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 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 is satisfied. Considering all the requirements, we can say that there is an α-morphism between N_C and N_I. Now that we proved that there is an α-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 some additional requirements which are stated as **c1**, **c2**, and **c3** in Section 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 Section 3, i.e., we consider the bubble in N_C and the 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

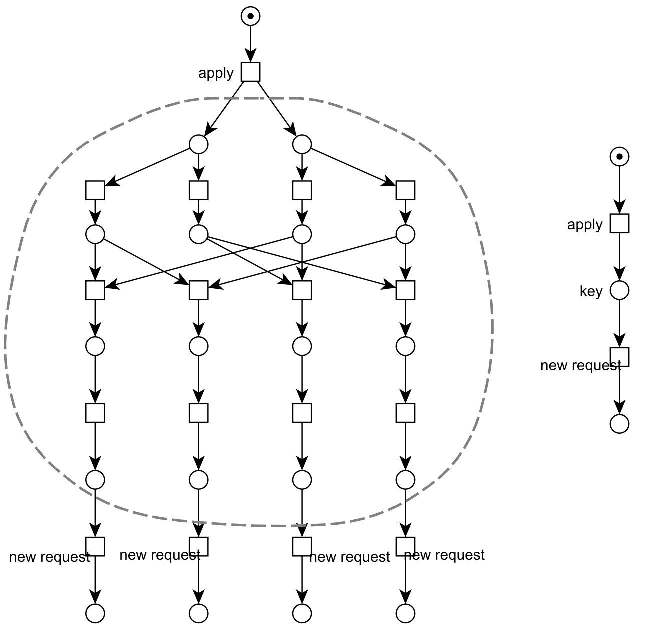

Fig. 7: $\mathit{Unf}(S_C(\mathit{key}))$ and $S_I(\mathit{key})$

$S_C(\mathit{key})$, represented as $\mathit{Unf}(S_C(\mathit{key}))$. The resulting nets are shown in Figure 7.

We follow the same procedure for N_{PKG} and we get two subnets $\mathit{Unf}(S_{PKG}$ $(\mathit{shares\ calculated}))$ and $S_I(\mathit{shares\,calculated})$ as in Figure 8.

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, so there is an α-morphism both from $\mathit{Unf}(S_C(\mathit{key}))$ to $S_I(\mathit{key})$ and from $\mathit{Unf}(S_{PKG}(\mathit{shares\,calculated}))$ to $S_I(\mathit{shares\,calculated})$. 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** are held. Moreover, considering Proposition 1, we can conclude that the composed net $N_{PKG}\langle N_I\rangle N_C$ is weakly bisimilar to N_I.

Knowing that our nets satisfy the requirements of α-morphisms and the other three additional constraints, give us the ensurance that, in addition to weakly bisimulation, the nets preserve another important property stated in **p1**. The property of reflecting reachable markings gives us a big advantage in performing reachability analysis. Instead of analyzing the big composed net with respect to reachability of a specific marking we can analyze the interface for the corresponding marking in the interface. 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

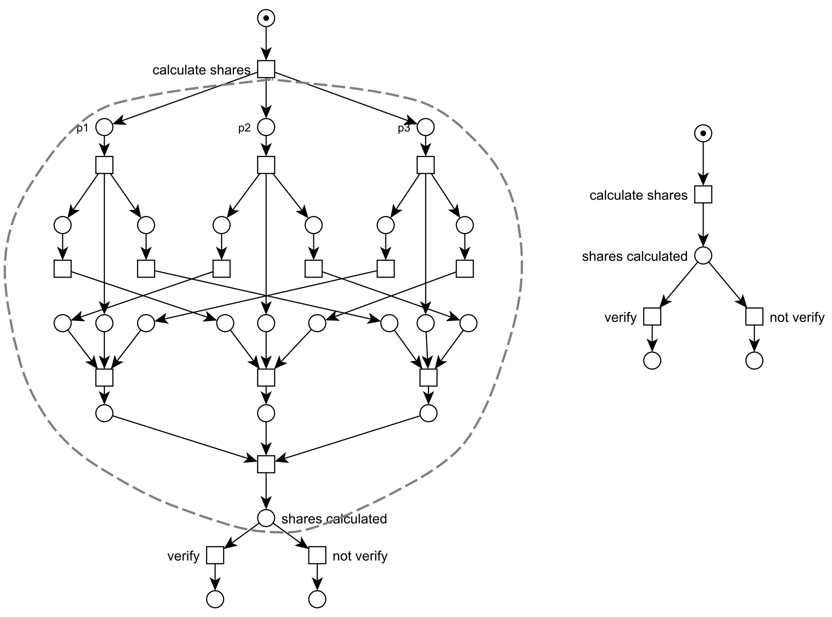

Fig. 8: $\mathit{Unf}(S_{PKG}(\mathit{shares\,calculated}))$ and $S_I(\mathit{shares\,calculated})$

condition *shares verified* should not be true while there is at least one token in any 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 interested global state 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 behavioural 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 [2] by means of a real world example. 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 PT 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.

References

1. Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Composition of Elementary Net Systems based on α-morphisms. In Michael Köhler-Bußmeier, editor, *Joint Proc. of LAM'12, WooPS'12, and CompoNet'12, Hamburg, Germany, June 25-26, 2012*, volume 853 of *CEUR Workshop Proceedings*, pages 87–102. CEUR-WS.org, 2012.
2. Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Local state refinement and composition on elementary net systems: an approach based on morphisms. Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), special issue based on the workshops of Petri nets 2012 (to appear), 2012.
3. Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Local State Refinement on Elementary Net Systems: an Approach Based on Morphisms. In Lawrence Cabac, Michael Duvigneau, and Daniel Moldt, editors, *Petri Nets and Software Engineering. International Workshop, PNSE'12, Hamburg, Germany, June 25-26, 2012. Proceedings*, volume 851 of *CEUR Workshop Proceedings*, pages 138–152. CEUR-WS.org, 2012.
4. Luca Bernardinello, Elena Monticelli, and Lucia Pomello. On preserving structural and behavioural properties by composing net systems on interfaces. *Fundam. Inform.*, 80(1-3):31–47, 2007.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32:586–615, 2003.
6. Aniket Kate and Ian Goldberg. Asynchronous distributed private-key generators for identity-based cryptography. *IACR Cryptology ePrint Archive*, 2009:355, 2009.
7. Gorkem Kilinc, Igor Nai Fovino, Carlo Ferigato, and Ahmet Koltuksuz. A model of distributed key generation for industrial control systems. In *11th Workshop on Discrete Event Systems*, volume 11, pages 356–363, Guadalajara, Mexico, 2012.
8. Elisabetta Mangioni. *Modularity for system modelling and analysis.* PhD thesis, Università degli Studi di Milano-Bicocca, Dottorato di ricerca in Informatica, ciclo 24, 2013.

9. Robin Milner. *Communication and concurrency.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
10. Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part i. *Theor. Comput. Sci.*, 13:85–108, 1981.
11. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theor. Comput. Sci.*, 96(1):3–33, 1992.
12. Lucia Pomello, Grzegorz Rozenberg, and Carla Simone. A survey of equivalence notions for net based systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 410–472. Springer, 1992.
13. Grzegorz Rozenberg and Joost Engelfriet. Elementary net systems. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 12–121. Springer, 1996.
14. A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology: Proceedings of CRYPTO 84, Lecture Notes in Computer Science*, 7:47–53, 1984.