A model of distributed key generation for industrial control systems

Gorkem Kilinc * **Igor Nai Fovino** ** **Carlo Ferigato** ***
Ahmet Koltuksuz ****

* *Izmir Institute of Technology, Izmir, Turkey (e-mail: gorkemklnc@gmail.com).*
** *Global Cyber Security Center GCSC, Rome, Italy (e-mail: igor.nai@gmail.com)*
*** *European Commission Joint Research Centre, Ispra, Italy (e-mail: carlo.ferigato@jrc.it)*
**** *Yasar University, Izmir, Turkey (e-mail: ahmet.koltuksuz@yasar.edu.tr).*

Abstract: The cyber-security of industrial control systems (ICS) is gaining high relevance due to the impact of industrial system failures on the citizen life. There is an urgent need for the consideration of security in their design, and for the analysis of the related vulnerabilities and potential threats. The high exposure of industrial critical infrastructure to cyber-threats is mainly due to the intrinsic weakness of the communication protocols used to control the process network. The peculiarities of the industrial protocols (low computational power, large geographical distribution, near to real-time constraints) make hard the effective use of traditional cryptographic schemes and in particular the implementation of an effective key management infrastructure supporting a cryptographic layer. In this paper, we describe a "model of distributed key generation for industrial control systems" we have recently implemented. The model is based on a known Distributed Key Generator protocol we have adapted to an industrial control system environment and to the related communication protocol (Modbus). To validate in a formal way selected security properties of the model, we introduced a Petri Nets representation. This representation allows for modeling attacks against the protocol and understanding some potential weaknesses of its implementation in the industrial control system environment.

Keywords: ICS – Industrial Control Systems; SCADA – Supervisory Control and Data Acquisition; security protocols; Petri Nets.

1. INTRODUCTION

The operative core of Industrial Control Systems (ICS) is generally known under the name of *Supervisory Control and Data Acquisition* — SCADA — system. The scope of this system is to gather data from the field devices (sensors), to define the proper reaction strategies and to send to the field actuators commands to maintain in operation the process of the industrial system. In this context, local control of sensors and actuators is realized via devices called *Programmable Logic Controllers* (PLC). These devices are sometimes spread over wide geographical areas, have relatively small computational power and can have tight time limits for guaranteeing a response to a command. Moreover, recent market constraints and engineering decisions have imposed the use of TCP/IP to these devices forcing the adaptation of the traditional serial protocols used for controlling the processes to a communication via TCP/IP packets. Consequently, attacks performed against Internet communication In Kilinc et al. (2012), a first step in the direction of obviating to the structural weakness of ICS protocols by adding a security layer to the transmitted packets was done. In this paper we aim at proving the effectiveness of the solution proposed and at opening our approach to the possibility of formal proofs of security properties. Moreover, we aim at considering completely the effective implementation of the protocol, possibly in the industrial environment in which it is used. To this scope, we decided to use as a modeling tool Elementary Net Systems for being able to describe at a very fine granularity the control flow in ICS. We subsequently extended the basic model to 1-bounded Petri Nets in order to allow for loops and use coherently a model checking tool for Petri Nets.

2. ICS SECURITY OVERVIEW

As claimed previously, modern critical industrial infrastructures (e.g. Power Plants, Water Grids etc.) use ICT technologies in an intensive manner.

Figure 1 provides a high level description of a typical industrial setting from an ICT point of view. In details figure 1.(a) shows the high level structure of different ICT networks involved in the process control. Figure 1.(b) provides an overview of the low level interaction in the control network between actuators, sensors and PLCs, while figure 1.(c) shows the interaction flows within the process network and the control network. As it is possible

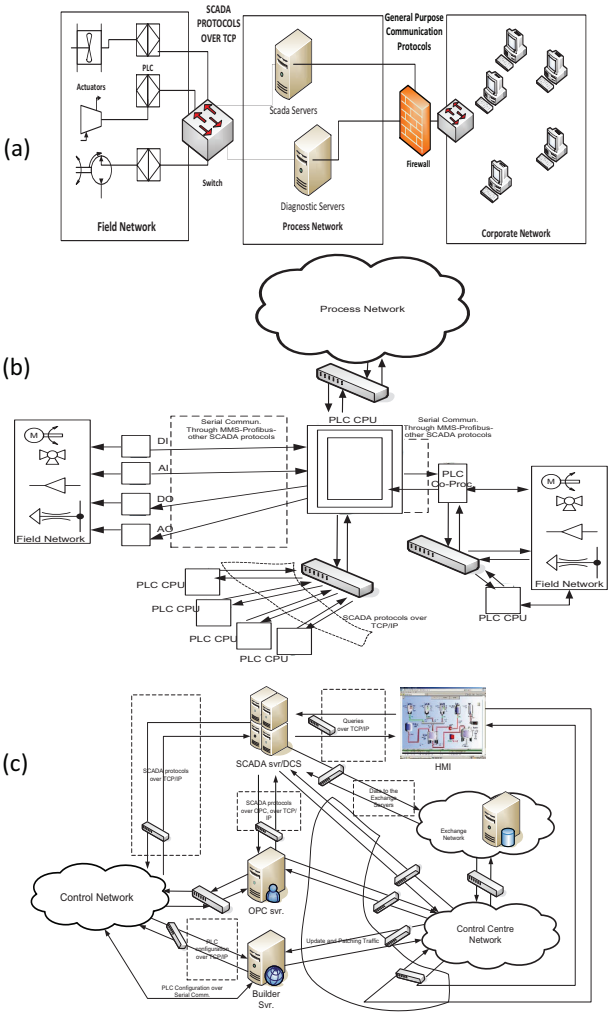

Figure 1. (a) High level schema of the SCADA cyber layer (b) control network (c) process network

to see, the core of the entire system is constituted by the control and data flows between the control network (i.e. the PLCs) and the process network (i.e. the SCADA servers).

Several studies (Dondossola et al. (2008); Carcano et al. (2010)) proved that modern industrial critical infrastructure are, on average, exposed to the traditional computer attacks and threats. Such an exposure is mainly due to the intrinsic weakness of the communication protocols used in SCADA systems to monitor and control the field devices.

Protocols like DNP3, Modbus, Profibus and Fieldbus were originally conceived for "serial communication" between SCADA master and slaves in a time in which there was no need for authentication, integrity and generally "ICT security" mechanisms.

In order to take advantage of the massive use of ICT, in the last 10 years several vendors ported those protocols over TCP/IP assuring the compatibility with the legacy application. As a result, nowadays the classical SCADA protocols are vulnerable to attacks which, in every other ICT context, would be considered out-of-date.

Only in the last years the research community started to pay attention to this problem. Adam and Byres presented an interesting high level analysis of the possible threats affecting a power plant system, a categorization of the typical hardware devices involved and some high level discussion about intrinsic vulnerabilities of the common power plant architectures in Creery and Byres (2005). A more detailed work on the topic of SCADA security, is presented in Chandia et al. (2007). In this work, the authors describe two possible strategies for improving the protection of SCADA networks, underlying that several aspects have to be improved in order to "secure" that kind of architectures. What is evident is that communication protocols used in such systems had not been conceived with security considerations in mind. Historically, this is due to the fact that when these protocols were designed, the world of industrial control systems was completely isolated from the public networks, and then ICT based intrusion scenarios were considered completely negligible. A variety of studies have been done about the security of such specialized communication protocols: for example, in M. Majdalawieh (2005) a proposal of extension of the DNP3 protocol which tries to address some of the known security problems of such master-slave control protocols (i.e. integrity of the commands, authentication, non repudiation etc.) is presented. Unfortunately, their work does not contain any detail on the implementation or test result, and for that reason it is hard to evaluate its efficacy. Similar approaches have been presented in Hong et al. (2007) while Mander et al. (2007) presents a proxy filtering solution aiming at identifying and avoiding anomalous control traffic. AGA (American Gas Association) proposed the AGA-12 standard in Consortium (2006), a cryptographic layer for serial SCADA communication. The use of IPsec or SSL/TLS to protect the DNP3 communication channel is proposed in Patel and Graham (2004).

The protocol which is presented in Bacnet (2009), implements some security capabilities; however, it is stated in Holmberg et al. (2003) that the authentication protocol implemented is vulnerable to man-in-the-middle attacks, parallel interleaving attacks and replay attacks. In Wright et al. (2004) a low latency encryption protocol based on CRC for retrofit SCADA link protection is presented. The work appears interesting for serial communications, however, the project does not seem to be updated since 2006. In Kilinc et al. (2012), finally a secure architecture for key generation and distribution suitable for enforcing the security of SCADA protocols is presented. In particular authors presented a prototype developed to enforce the Modbus over TCP protocol. Even if different other implementation of Modbus exist, it is evident how the characteristics of TCP are the most suitable for a reliable implementation of the Distributed Private Key Generation (DPKG) architecture. This paper takes start from this last work, with the aim of proposing a formal model to be used to validate the security properties guaranteed by Kilinc et al. (2012). To define this model we adopted the Petri Net paradigm. In the following section a brief introduction to the related Petri Net literature is presented.

3. PETRI NETS AND SECURITY PROTOCOLS

Given the possibility to represent, at the lowest level, the communication flow and for their intuitive graphical presentation, Petri Nets are used ever since for the detailed

analysis of communication protocols. Starting from the work of Nieh and Tavares (1992), Petri Nets have been applied to the analysis of security protocols as well. While in Nieh and Tavares (1992) the construction of the model is based on a refinement process involving P/T net systems for the *conceptual level* and high-level nets for the *functional level*, in more recent works, almost exclusively high-level nets have been used. For example, in Xu and Xie (2011), Colored Petri Nets are used for re-discovering known flaws in the *Andrew secure RPC protocol* while in Bouroulet et al. (2008), a special class of composable high-level Petri Nets is used for showing flaws in the *Kao-Chow authentication protocol*. In all of the cases, model checking techniques are applied to the protocol model in order to generate the cases witnessing the flaw in the protocol.

In this study, a different approach to model a security protocol is chosen since the application domain, ICS, is quite specific and low level control flow should be represented. Consequently, we decided to model the protocol with low-level nets starting with Elementary Net Systems and subsequently extending it to 1-bounded P/T systems. In this way, we have at least three advantages: 1) leave the representation open to the inclusion of low-level signal processing; 2) use basic tools for proving structural properties of the model like invariants analysis at the level of basic signals; 3) construct easily abstractions of the model by passing to higher-level nets. Moreover, this choice allows for the use of powerful model checking tools like LoLA, Schmidt (2003).

4. THE DPKG MODEL

A net N is a bipartite, non-empty, directed graph $N = (P, T, F)$. P and T are the nodes of the graph. The P elements are called *places* (drawn as circles), the T elements are called *transitions* (drawn as boxes) and $F \subseteq (P \times T) \cup (T \times P)$ — the set of edges of the graph — is called *flow relation*. It is assumed that the net has no isolated nodes.

If $X = P \cup T$ and $x \in X$, ${}^\bullet x$ denotes the set $\{y \in X \mid (y, x) \in F\}$ while $x^\bullet$ denotes the set $\{y \in X \mid (x, y) \in F\}$.

States of Petri Nets are defined by markings, that is mappings $m : P \longrightarrow \{0, 1, 2, \ldots\}$. A transition $t \in T$ is *enabled* at a marking m if, for all $p \in {}^\bullet t$, $m(p) > 0$. In this case, t can occur and the occurrence of t transforms m into a new marking m':

$$m'(p) = \begin{cases} m(p) - 1 & \text{if } p \in {}^\bullet p \setminus p^\bullet; \\ m(p) + 1 & \text{if } p \in p^\bullet \setminus {}^\bullet p; \\ m(p) & \text{otherwise.} \end{cases}$$

A marking m is graphically represented by n black *tokens* in place p if $m(p) = n$. A net N endowed by an initial marking m_{in} is called *marked net*; Desel and Reisig (1998).

It is possible to prove that, in the marked nets we are considering, all markings m' reachable from the initial marking are such that $m' : P \longrightarrow \{0, 1\}$. In other words, the net systems we are considering are *1-bounded*.

In an id-based cryptographic system, unlike 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. Shamir's scheme presented in Shamir (1979) enables users to communicate securely and 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. PKG has a key pair which is referred as master key pair consisting of a master private key and a master public key. 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, 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 is one of the effective solutions to the key escrow problem. In both schemes Boneh and Franklin (2003), Kate and Goldberg (2009) 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 by Boneh and Franklin (2003) and Kate and Goldberg (2009) and contributed by providing a formal model and analysis by Petri Nets. For more detail about the algorithms and the terminology it is recommended to refer to Boneh and Franklin (2003), Kate and Goldberg (2009) and Kilinc et al. (2012).

The identity based distributed private key generation scheme is modeled in two parts as distribution and extraction.

The net shown in figure 2 represents the distribution step of $(3, 2)$ distributed private key generation scheme which means that there are three private key generators (PKG) and the key can be constructed only if at least two pieces of the key are acquired from two distinct PKG nodes.

The initial marking assigns to the places with labels *idle*, *initCanStart* and *BBidle* one token each, while the other places have no tokens. Since the net system is 1-bounded, it is possible to interpret *idle*, *initCanStart* and *BBidle* as logically true local conditions while all the other local conditions are logically false. The only event that can be fired is *initializePKGnodes*.

After initializing PKG nodes, PKG1, PKG2 and PKG3 are ready. From this point on, there are three flows (subnets) representing three PKG nodes. The fourth subnet represents the Bulletin Board. Only one of the flows will be explained here since the other two are alike. The transition *chooseP1andSendToBB* means PKG1 chooses a polynomial whose coefficients are random elements from a predefined field. After choosing the polynomial P1, PKG1 also sends the commitments $C_{1l} = a_{1l}g$ for $l = 0, 1, 2$ to the Bulletin Board to be broadcasted. a_{il} is the l^{th} coefficient of i^{th} polynomial. It is important to note that

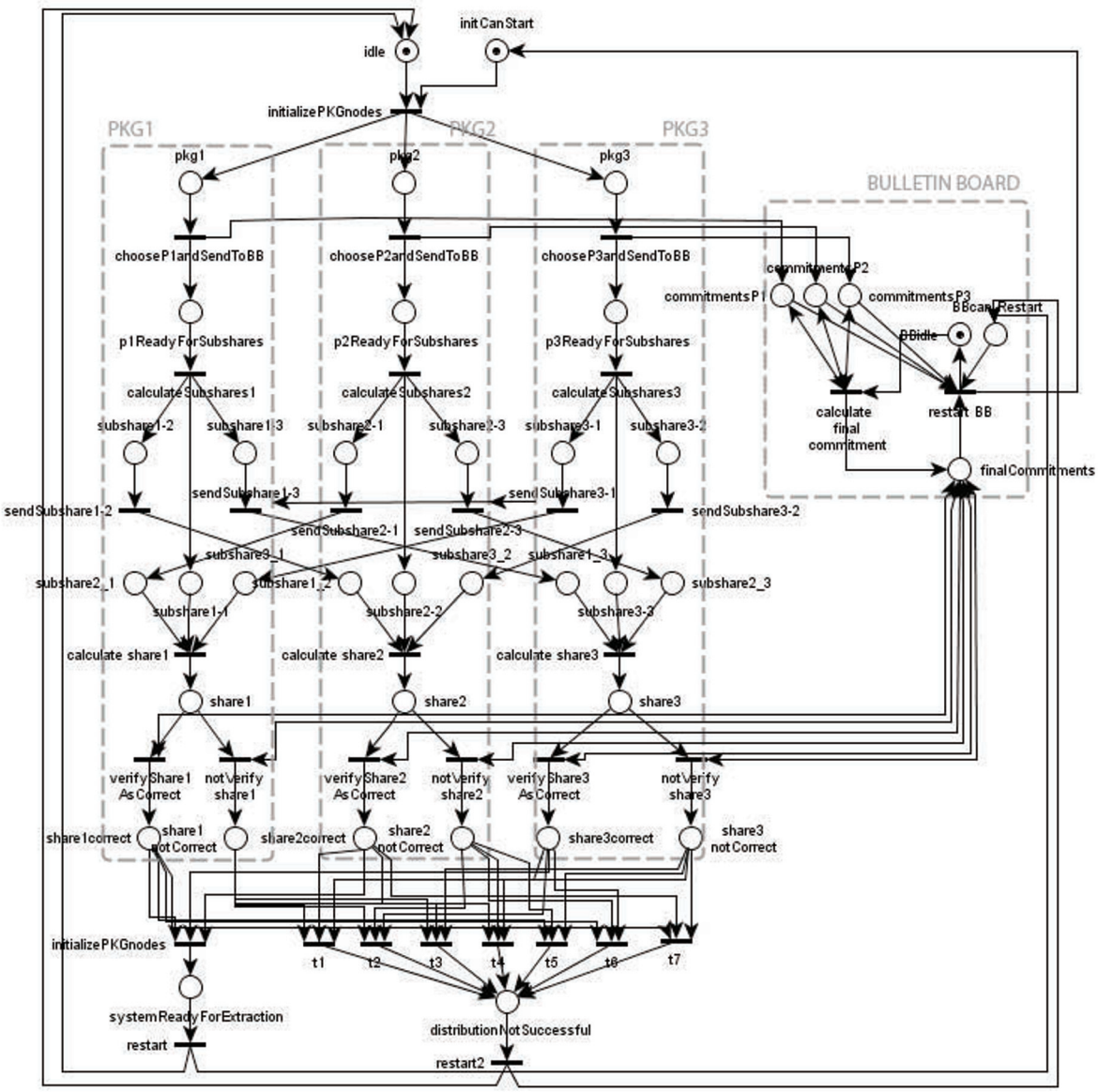

Figure 2. Petri Net model of distribution step of distributed private key generation.

PKG1 does not send the polynomial; it only sends the commitments. When this transition is fired, the local states *P1readyForSubshares* and *commitmentsP1* become true.

Independently, when all three PKG nodes send the commitments, all four places in the preset of *calculate final commitment* transition become true so the transition is enabled. After *p1ReadyForSubshares* becomes true, the transition *calculateSubshares1* is enabled. The firing of this transition leads to the case that three subshares of P1 (*subshare1-1*, *subshare1-2*, *subshare1-3*) are calculated and these three local states become true, while the token of *p1ReadyForSubshares* is consumed. One of the subshares of each PKG node is for itself while the other two are to send to other PKG nodes.

A PKG node needs three subshares including its own to continue with calculating its share for the master private key. Thus, PKG1 needs PKG2 to send *subshare2-1* and PKG3 to send *subshare3-1*. If related transitions are already fired and *subshare 2_ 1*, *subshare1-1* and *subshare3_ 1* local states are all true, The transition *calculate share1* is enabled. Note that the local states *subsharei-j* and *subsharei_ j* are different places. *subsharei_ j* represents the subshare which is sent to PKGj by PKGi, while *subsharei-j* is the subshare calculated by PKGi to send the PKGj but still being held in PKGi.

After *share1* becomes true, if final commitment is calculated, two transitions are enabled: *verifyShare1AsCorrect* and *notVerifyShare1*. Each PKG node follows the same flow and verifies their shares by using the final commitment which is held by Bulletin Board. If all the PKG nodes verify their shares as correct, then the system is ready for the extraction. Otherwise, there can be no extraction and the system needs to restart. If at least one of the shares is incorrect then one of the transitions *t1,t2,t3,t4, t5, t6, t7* becomes enabled. Firing of this transition consumes the remaining tokens to make system ready to restart.

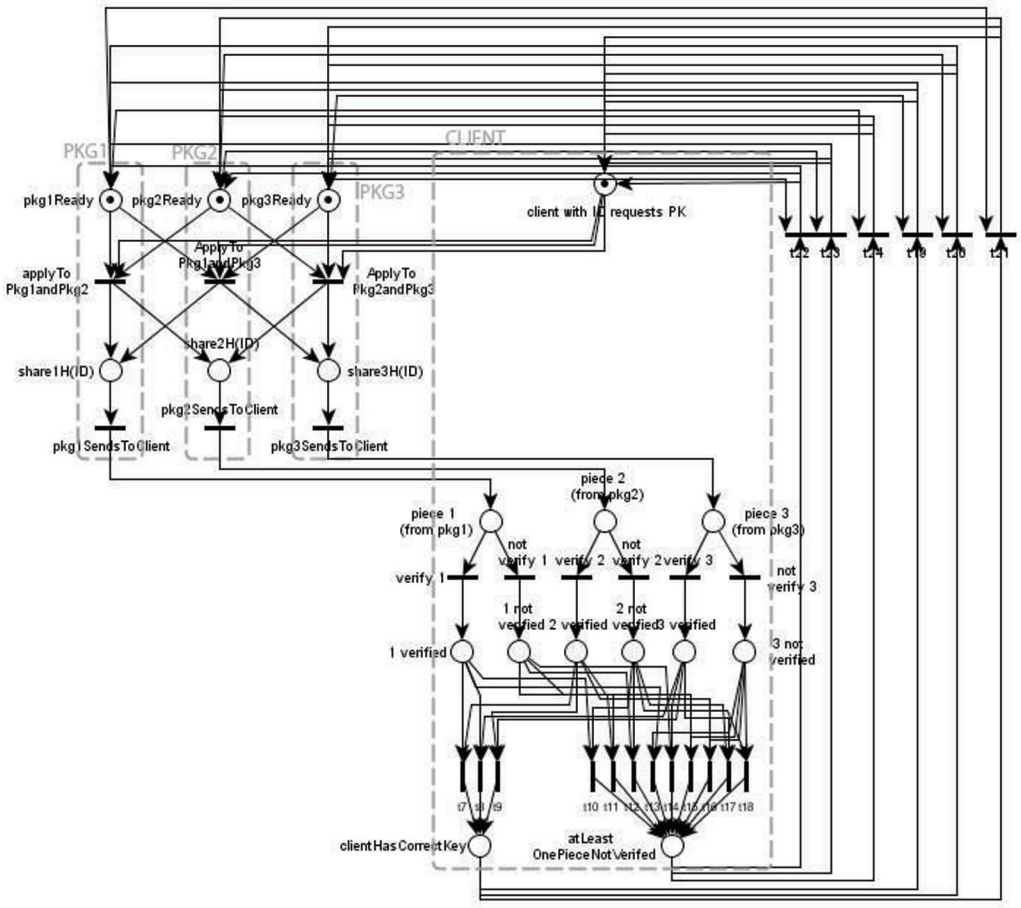

Figure 3. Petri Net model of extraction step of distributed private key generation.

Only one of the local states *systemReadyForExtraction* and *distributionNotSuccessful* can be true and the system can restart both after a success and fail. For each restart case, Bulletin Board also becomes ready to restart. The place *BBCanRestart* eliminates deadlock situations like the final commitment is still being used but Bulletin Board restarts and there is no final commitment. *InitCanStart* ensures that Bulletin Board restarts before initialization of the PKG nodes. It is necessary to clear the previous commitment values from the Bulletin Board and make the system ready for the new distribution.

The Petri Net model of the extraction part is shown in figure 3. Extraction starts after all three shares are ready and verified as correct and there is a client requesting a private key. The places *pkg1Ready*, *pkg2Ready* and *pkg3Ready* in extraction model overlaps with *share1correct*, *share2correct* and *share3correct* in the distribution model.

One can easily notice that the whole model is divided into two parts as distribution of the shares and the extraction of the client's private key in a way that two nets can overlap. This kind of division provides us with a more detailed and powerful modeling capability and a possibility to analyze the models without having memory and time constraint problems.

We have stated that the models represent a $(3, 2)$ scheme which means has to apply to two of the PKG nodes because in this $(3, 2)$ scheme private key can be constructed by using two shares from two distinct PKG nodes. When the system starts, there is a client, whose ID is known by every node, waiting for a private key and three nodes with three shares are ready for an extraction.

All of the three transitions (*applyToPkg1andPkg2*, *applyToPkg1andPkg3* and *applyToPkg2andPkg3*) are enabled but only one can be fired.

For instance if *applyToPkg1andPkg2* transition is fired, *share1H(ID)* and *share2H(ID)* become true. The firing of the transition actually means that PKG1 does elliptic curve scalar multiplication with share1 and hash value of client's identity string, H(ID). PKG2 does the same. In the net sending the result to the client is done by transitions *pkg1SendsToClient* and *pkg2SendsToClient*.

After sending, the flow goes on in the client part. In this case, *piece1* and *piece2* states become true. For each place representing a piece there are two transitions as *verify* or *not verify*. Both of the pieces need to be verified so that client can use the pieces and calculate its private key. If any of the pieces is not verified, the system reaches an unsuccessful state and the local state *atLeastOnePieceNotVerified* becomes true. After both successful and unsuccessful states the system can be restarted for a new extraction. There are six restart transitions since each one is enabled in a different ending state and the old tokens have to be consumed in order to reproduce the initial marking.

5. ANALYSIS ON THE DPKG MODEL

In this section, we introduce a structural analysis of the DPKG model for Identity based Cryptography. The purpose of modeling the protocol is not only to provide a formal representation of the system but also a basis for formal analysis of it. Petri Net model enables us to do a security property analysis for the protocol.

The first basic analysis on the models in figure 2 and figure 3 shows that the nets are both bounded both safe and they are deadlock free. The analysis result tells us that the system will go on working safely unless there is a physical, electrical, human error problem, etc. The model supports the fact that DPKG can perform repeatedly whenever there is a need for (1) changing the master key and recreate shares for PKGs , (2) extracting the same private key for the client in case it has lost its key or (3) extracting a new private key for a new client who has just joined the system.

For a more specified analysis we move to the model checking environment and use the model checking tool LoLA. To express the properties or situations needed to be analyzed, we first defined the situation as a global states of the model and translated it into temporal logic formulas expressed in CTL (Computation Tree Logic).

FORMULA: EXPATH EVENTUALLY (finalCommitments = 0 AND (systemReadyForExtraction = 1 OR distributionNotSuccessful = 1))

The above formula investigates the existence of a global state in which PKG nodes finish the distribution of subshares and calculating shares before Bulletin Board calculates final commitments. This is an unwanted situation which means that the verification of the shares is not performed and the calculated shares cannot be trusted.

When we check the model in figure 2, we see that the result for the formula is false which means the global state is not reachable in the model.

FORMULA EXPATH EVENTUALLY systemReadyForExtraction = 1 AND distributionNotSuccessful = 1

The second formula focuses on the possibility of a conflict in the end of the distribution step. It asks if there exists a global state in which the success and the fail states are true in the same time. The result of the formula is false when the model in figure 2 is checked for the formula.

FORMULA EXPATH EVENTUALLY (clientHasCorrectKey = 1 AND atLeastOnePieceNotVerified = 1)

The last formula checks the existence of a global state in the extraction model shown in the figure 3. This global state defines the case in which client has the correct key but at least one of the pieces is not verified. When the model is checked for the formula, it is proved that the global state is not reachable.

6. ATTACK MODEL

We introduce an intruder model which sneaks into the channel during the distributed key generation. This model is also composed of two parts: distribution and extraction. The first intruder model is shown in figure 4. In this model we included only two PKG nodes to keep the model small but sufficient enough to analyze the properties we are interested in like what can be done by the intruder in such a scenario. The intruder listens to the channel between PKG1 and PKG2 having also the possibility to change the message on the channel. When PKG1 sends the subshare for PKG2, *subshare1-2*, the intruder has two options. It can read *subshare1-2* and do no change on it or it can send a fake *subshare1-2* to PKG2 while keeping the original one to itself. The same flow exists also in the opposite direction to read and change the *subshare2-1*. The modeled DPKG here is (2,1) which means there are two PKG nodes and in the distribution step they produce two subshares.

The second intruder model representing the extraction step with an intruder, is shown in the figure 5. In this model there are three PKG nodes and the system is (3,2). This is the smallest sufficient and meaningful model for analyzing the intruder behavior. In this model, intruder listens to the channels between three PKG nodes and the client. The intruder can read the piece sent by the PKG node to the client and then can change the message or leave it as it is. Another option is that the intruder does not read anything from the channel. In this way, we model an intruder who can choose the channel to be listened and has the ability to change the messages on the channels.

7. ANALYSIS ON THE ATTACK MODEL

After creating the model we used model checking with LoLA to test the reachability of specific global states threatening the system. To express the unwanted situations we first defined the situation as a global states of the model and translated it into temporal logic formulas expressed in CTL as we did for the DPKG analysis in section 5.

FORMULA: EXPATH EVENTUALLY ((subshare1-2AtIntruder = 1 AND subshare2-2AtIntruder = 1) OR (subshare2-1AtIntruder = 1 AND subshare1-1AtIntruder = 1))

This formula investigates if it is possible that there exists a global state in which the intruder captures both subshares for PKG1 or PKG2. When the model displayed in figure 4 is checked for this formula, the result is found to be false. In an (n,k) system, to be able to have a PKG node's share, an intruder has to capture n subshares addressing that specific PKG node. But this situation cannot occur, because one of these subshares is created by the node itself and it is never sent to the channel.

FORMULA EXPATH EVENTUALLY (clientHasCorrectKey = 1 AND atLeastOnePieceNotVerified = 1)

The above formula investigates a situation in which the client supposes to have the correct key while at least one of the received pieces is not verified. When the model shown in figure 5 is checked for this formula we see that the result is false.

FORMULA: EXPATH EVENTUALLY (intruderHasTheKey = 1 AND clientHasCorrectKey = 1)

The last formula asks if it is possible for the intruder to have the client's key without the client being informed

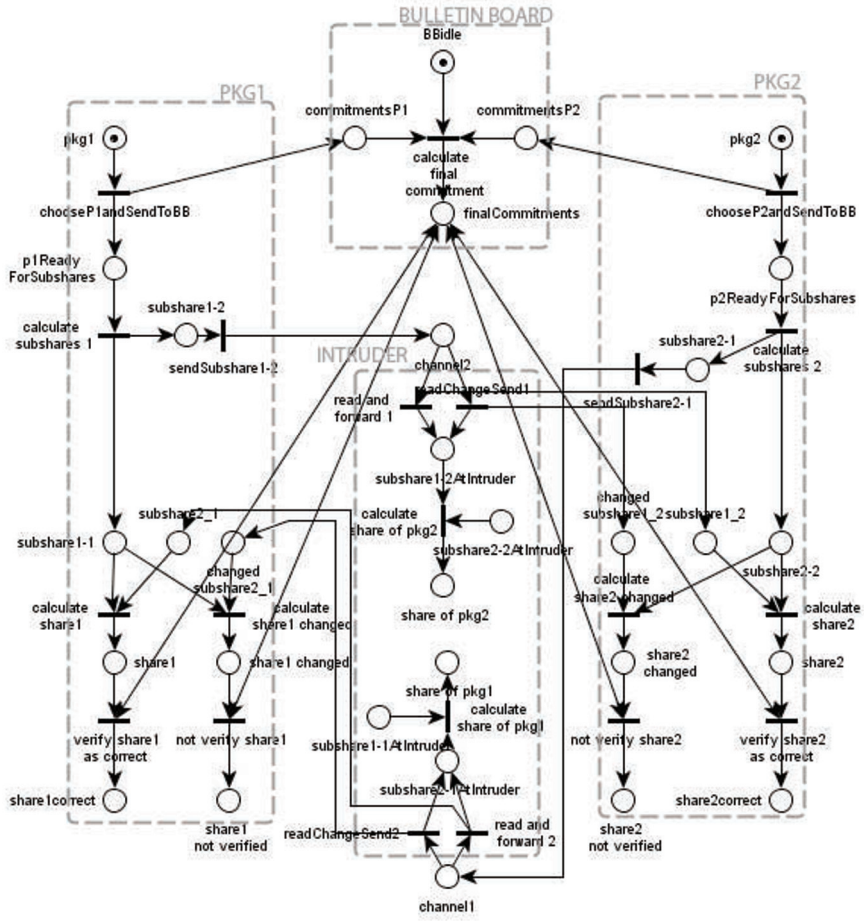

Figure 4. Petri Net model of an intruder in the distribution step of DPKG.

about it. This situation is likely to occur if the intruder captures all the pieces needed for extraction of the client's key and does no change on them so that client will also have the same key. The number of required pieces is k in an (n,k) system. After checking the model for this formula, the result turns out to be true. However, this situation is a natural outcome for threshold cryptographic protocols such as DPKG for identity based cryptography.

8. CONCLUSION

We aimed to analyze the behavior of the DPKG security protocol applied to an industrial control system and proving its properties in the specific context of application.

To this end, a basic model of the protocol is built by using Elementary Net Systems and then extended to 1-bounded P/T systems. This model allows for the first reflections about the protocol by means of traditional structural analysis techniques. Then we extended our work by including an intruder in the model aiming at analyzing the security properties of the protocol and revealing possible flaws. The results indicate that there is no way to extract information about the master key or the private keys of the clients. Thus it can be concluded that the models drawn and the analysis performed on them have not pointed out any flaws except the drawbacks of the threshold cryptography. In this way, use of the protocol in the industrial control systems is partially proved to be safe. An attacker is not able to gain any information about the system and two main goals of an attacker which are stated in the beginning of the section 6 are not to be achieved. Our work only focuses on the protocol of distributed private key generation, thus we cannot say anything about the use of the keys for encryption, decryption and signature.

In addition, our models are given in an abstract and theoretical manner and our future work will aim to extend the model with the inclusion of local acquisition and actuation devices (PLCs). Such a study will allow for a real representation of the protocol in its environment and the subsequent possibility of analyzing its real flaws with model checking techniques. The other extension will be to model the use of the generated keys in the signature scheme to provide industrial control systems with authenticity. In this way we will be able to formally analyze the authenticity in the system.

REFERENCES

Bacnet (2009). URL `http://www.bacnet.org/`.

Boneh, D. and Franklin, M.K. (2003). Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32, 586–615.

Bouroulet, R., Devillers, R.R., Klaudel, H., Pelz, E., and Pommereau, F. (2008). Modeling and analysis of security protocols using role based specifications and petri

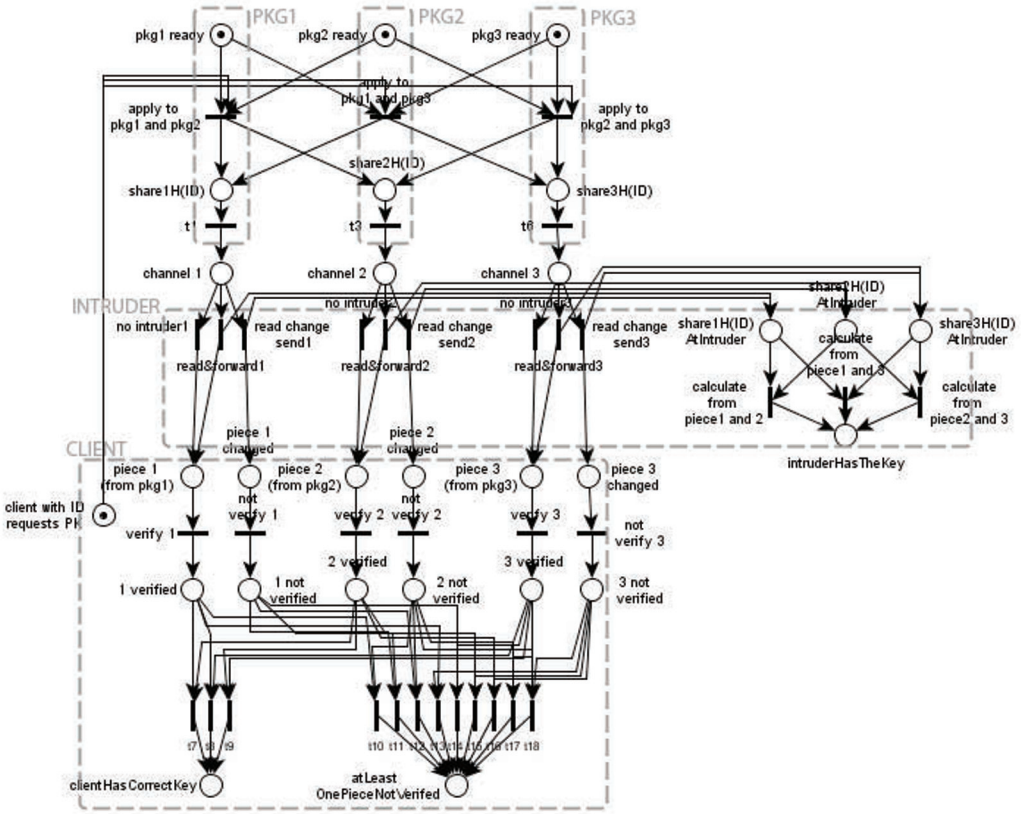

Figure 5. Petri Net model of an intruder in the extraction step of DPKG.

nets. In *Petri Nets*, volume 5062 of *Lecture Notes in Computer Science*, 72–91. Springer.

Carcano, A., Fovino, I.N., Masera, M., and Trombetta, A. (2010). State-based network intrusion detection systems for scada protocols: a proof of concept. In *Proceedings of the 4th international conference on Critical information infrastructures security*, CRITIS'09.

Chandia, R., González, J., Kilpatrick, T., Papa, M., and Shenoi, S. (2007). Security strategies for scada networks. In *Critical Infrastructure Protection*, volume 253 of *IFIP*. Springer.

Consortium, A. (2006). URL `http://www.aga.org/`.

Creery, A.A. and Byres, E.J. (2005). Industrial cybersecurity for power system and scada networks. *IEEE Industry Apllication Magazine*, 13, 303–309.

Desel, J. and Reisig, W. (1998). Place/transition petri nets. In *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*. Springer.

Dondossola, G., Szanto, J., Masera, M., and Nai Fovino, I. (2008). Effects of intentional threats to power substation control systems. *International Journal of Critical Infrastructures*, 4, 129–143.

Holmberg, D.G., Holmberg, D.G., and Evans, D.L. (2003). Bacnet wide area network security threat assessment.

Hong, J.H.C.S., Ju, S.H., Lim, Y.H., Lee, B.S., and Hyun, D.H. (2007). A security mechanism for automation control in plc-based networks. In *ISPLC '07. IEEE International Symposium on Power Line Communications and Its Applications*.

Kate, A. and Goldberg, I. (2009). Asynchronous distributed private-key generators for identity-based cryptography. *IACR Cryptology ePrint Archive*, 2009, 355.

Kilinc, G., Fovino, I.N., Ferigato, C., and Koltuksuz, A. (2012). A model of distributed key generation for industrial control systems. Jrc technical note 69663, E.C. Joint Research Centre, Institute for the Protection and Security of the Citizen.

M. Majdalawieh, F. Parisi-Presicce, D.W. (2005). Distributed network protocol security (dnpsec) security framework. In *21st Annual Computer Security Applications Conference*.

Mander, T., Nabhani, F., Wang, L., and Cheung, R. (2007). Data object based security for dnp3 over tcp/ip for increased utility commercial aspects security. In *Power Engineering Society General Meeting, IEEE*.

Nieh, B.B. and Tavares, S.E. (1992). Modelling and analyzing cryprographic protocols using petri nets. In *AUSCRYPT*, volume 718 of *LNCS*. Springer.

Patel, S. and Graham, J. (2004). Security considerations in dnp3 scada systems. In *International Conference on Computer Applications in Industry Engineering*.

Schmidt, K. (2003). Distributed verification with lola. *Fundam. Inform.*, 54, 253–262.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22, 612–613.

Wright, A.K., Kinast, J.A., and McCarty, J. (2004). Low-latency cryptographic protection for scada communications. In *ACNS*, volume 3089 of *Lecture Notes in Computer Science*. Springer.

Xu, Y. and Xie, X. (2011). Modeling and analysis of security protocols using colored petri nets. *JCP*, 6, 19–27.