YAŞAR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A PRACTICAL AND FORMAL ANALYSIS OF DISTRIBUTED PRIVATE KEY GENERATION FOR IDENTITY BASED CRYPTOGRAPHY

Görkem KILINÇ

Thesis Advisor: Assoc. Prof. Ahmet Hasan KOLTUKSUZ, Ph.D

Department of Computer Engineering

Bornova-İZMİR
June 2012

ÖZET

KİMLİK TABANLI KRİPTOGRAFİK SİSTEMLER İÇİN DAĞITIK GİZLİ ANAHTAR ÜRETİMİNİN İŞLEVSEL VE BİÇİMSEL AÇIDAN İNCELENMESİ

KILINÇ, Görkem

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Doç. Dr. Ahmet Hasan KOLTUKSUZ

Haziran 2012, 77 sayfa

Temelleri gizlilik paylaşımı yöntemlerine dayanan dağıtık gizli anahtar üretimi(DGAÜ) kriptografide gizli anahtarların dağıtık olarak güvenli bir şekilde üretilmesi amacıyla kullanılmaktadır. DGAÜ kullanımına en uygun olan kriptografik sistemlerden bir tanesi kimlik tabanlı kriptografik sistemdir. DGAÜ kullanımı, kimlik tabanlı kriptografik sistemlerde var olan ve sistemin pratik olarak kullanımında engel teşkil eden anahtar emaneti probleminin bir çözümü olarak bilinmektedir. Her ne kadar bilinen matematiksel problemler üzerine oturmuş olsa da, öngörülmeyen açıklar içerebileceği için protokolün doğrulanması çok önemlidir. Bu tezde, DGAÜ protokolünün hem işlevsel hem de biçimsel açıdan analizi hedeflenmiştir. Bu amaçla öncelikle DGAÜ protokolü, endüstriyel kontrol sistemleri üzerinde büyük ölçüde kullanılmakta olan SCADA protokolünde eksik olan kimlik denetimi sağlanarak güvenli hale getirilmesi için uygulanmıştır ve işlevsel analiz bu uygulama üzerinde yapılmıştır. Biçimsel (formal) analiz için ise protokol basit Petri ağları kullanılarak modellenmiştir. Daha sonra bu model üzerinde yapısal ve davranışsal analizler yapılarak protokolün biçimsel olarak doğrulanması için gerekli güvenlik özellikleri ispatlanmıştır.

Anahtar Kelimeler: Dağıtık Gizli Anahtar Üretimi (DGAÜ), Kimlik Tabanlı Kriptografi, Endüstriyel Kontrol Sistemleri, SCADA, Basit Petri Ağları, Biçimsel (Formal) Analiz, Kriptografik Protokollerin Biçimsel (Formal) Doğrulaması.

ABSTRACT

A PRACTICAL AND FORMAL ANALYSIS OF DISTRIBUTED PRIVATE KEY GENERATION FOR IDENTITY BASED CRYPTOGRAPHY

KILINÇ, Görkem

M.Sc. in Computer Engineering

Supervisor: Assoc. Prof. Ahmet Hasan KOLTUKSUZ, Ph.D

June 2012, 77 pages

The basis of distributed private key generation (DPKG) scheme lies on secret sharing and it has been used in cryptography for generating keys distributively and securely. One of the most suitable cryptographic system for DPKG is identity based cryptographic system. DPKG is enormously used for generating private keys for the clients of an identity based cryptographic system. The scheme is claimed to be a solution for the key escrow problem which exists in the identity based cryptography and leads to impractical cryptographic systems. It is highly important to verify the protocol even though the protocol lies on well known mathematical problems. Clearly, the protocol might always preserve some unforeseen flaws. In this thesis, we aimed to analyze DPKG protocol in both practical and formal manners. For this aim, we implemented the protocol on industrial control systems (ICS) in order to secure the widely used ICS protocol SCADA by providing it with authentication and performed a practical analysis on the implementation. In order to perform formal analysis, we first modeled the DPKG protocol by using elementary Petri nets then we performed structural and behavioral analysis on the model for proving essential security properties of the protocol with the eventual aim of formal verification.

Keywords: Distributed Private Key Generation (DPKG), Identity Based Cryptography, Industrial Control Systems, SCADA, Elementary Petri Nets, Formal Analysis, Formal Verification of Cryptographic Protocols.

Acknowledgements

First and foremost, a very special thanks is due to my supervisor, Assoc. Prof. Ahmet Koltuksuz, Ph.D for his invaluable guidance, patience and encouragement. I feel very fortunate for having had the chance to work with him. He is such an exceptional advisor that I will always be honored to be his student.

I also want to present my sincere thanks to European Commission Joint Research Centre (JRC), Ispra. An important part of the work has been done during the year I spent there as a trainee. Many thanks for the opportunity.

I would like to thank Carlo Ferigato, Ph.D who introduced me with Petri nets and Igor Nai Fovino, Ph.D. They had been devoted advisors to me during my traineeship in JRC. I am enormously thankful to them for their still-continuing support from overseas.

I owe my deepest gratitude to Asst. Prof. Serap Şahin, Ph.D who has always stood by me with her never ending support, guidance and positive energy. She was the one who uplifted me during the hard times.

Many thanks go to Asst. Prof. Hüseyin Hışıl, Ph.D for his huge support. He helped me study abstract algebra, elliptic curves and cryptographic pairings which formed an important part of the mathematical basis for my study.

I also want to express my appreciations to my dear friends and classmates Burak Ekici and Çağatay Yücel. For many years we have studied together and they have been always very helpful and supportive.

Last but not the least I owe more than thanks to my mother Şengül Kılınç, my father Hasan Kılınç, and two very special and valuable people in my life; Egemen Soylu and Burcu Ezgi Özdemir. They have always believed in me and supported me. Without them I might never have finished this thesis.

TEXT OF OATH

I declare and honestly confirm that my study titled “A Practical and Formal Analysis of Distributed Private Key Generation for Identity Based Cryptography”, and presented as Master’s Thesis has been written without applying to any assistance inconsistent with scientific ethics and traditions and all sources I have benefited from are listed in bibliography and I have benefited from these sources by means of making references.

11 / 06 / 2012

Görkem KILINÇ

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Related Work

In 1984, Adi Shamir [1] put forward the idea of identity based cryptography. This cryptographic scheme was not very practical because of the need for a private key generator which is responsible for generating keys for the users. Generating private keys for the user, private key generator can sign or decrypt a message on behalf of any user and it can make users' private keys public. This problem of identity based cryptography is called key escrow problem. A solution to the key escrow problem was proposed by Boneh and Franklin [2] in 2001. Their solution was distributing the private key generation among multiple private key generators.

On the other side another concept was being developed which is called Petri nets. The concept was first defined by Carl Adam Petri in his Ph.D. thesis in 1960 [3]. Petri net is a powerful modeling tool especially for distributed, concurrent, asynchronous systems. It is intensively used for modeling communication and in the last years it is being used for analysis and verification of cryptographic protocols as well. However there is no known Petri net model for distributed private key generation.

Considering all, the aim of the work is to provide a formal model of distributed private key generation (DPKG) for identity based cryptography, to analyze the structural and behavioral properties, to use model checking methods on the model for more detailed analysis and finally to verify some security properties of DPKG to make a step for the verification of the protocol.

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 [4], Petri Nets have been applied to the analysis of security protocols as well. While in [4] the construction of the model is based on a refinement process involving Place/Transition 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 [5], Colored Petri Nets are used for re-discovering known flaws in the *Andrew secure RPC protocol* while in [6], 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, Information and Communication Security (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. 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 abstractions of the model easily by passing to higher-level nets. Moreover, this choice allows for the use of powerful model checking tools like LoLA [7].

1.2 Roadmap

The rest of the thesis is structured as follows.

Chapter 2 provides operational description of identity based cryptography and distributed private key generation (DPKG) after giving a brief mathematical background.

Chapter 3 presents a practical specification of DPKG for identity based cryptography. The chapter includes an implementation of DPKG on industrial control systems. In the implementation DPKG is used for generating identity based private keys then these keys are used for performing identity based signature in the aim of providing the system with authentication.

In *Chapter 4*, after an introduction to Petri nets, the definition and properties of elementary Petri nets are given. In the end of the chapter we discuss how elementary Petri nets are used for modeling and analyzing cryptographic protocols.

In *Chapter 5*, Petri net models of DPKG are introduced. First, a basic model of the protocol is presented and then an intruder is added into the model. In addition to traditional structural analysis, model checking methods are also used to prove security properties of the protocol.

Chapter 6 discusses the results of the study and concludes the thesis.

Chapter 2

Identity Based Cryptography and Distributed Private Key Generation

2.1 Mathematical Background

2.1.1 Preliminaries

Definition 2.1. A **group** is a pair $(G, *)$ where G is a nonempty set closed under the binary operation $*$, such that the following axioms are satisfied:

- $*$ is associative; $(P * Q) * R = P * (Q * R)$, where $P, Q, R \in G$
- There is an identity element e such that; $e * P = P * e = P$ where $e, P \in G$
- Each element Q in G has a unique inverse Q' in G with the property that; $Q * Q' = Q' * Q = e$

Definition 2.2. A group G is an **abelian group** if its binary operation $*$ is commutative.

Example 2.1. The set $\mathbb{Z}^+$ is not a group, because there is no identity element in $\mathbb{Z}^+$.

Example 2.2. The set of all nonnegative integers $\mathbb{Z}^+ \bigcup \{0\}$ is still not a group. It has an identity element which is 0 but not all the elements in the set has an inverse.

Example 2.3. The set of integers $\mathbb{Z}$, rational numbers $\mathbb{Q}$, real numbers $\mathbb{R}$ and complex numbers $\mathbb{C}$ with the operation + are abelian groups.

Definition 2.3. If G is a finite group, the **order** of G, denoted by $|G|$, is the number of elements in G.

Definition 2.4. A subset H is a **subgroup** of G, if H is closed under the binary operation of G and if H is a group itself.

Theorem 2.1. Let G be a group and $P \in G$. Then,

- $H = \{P^n \mid n \in \mathbb{Z}\}$ is a subgroup of G.
- H is the smallest subgroup of G which contains P.
- Every subgroup containing P contains H.

Proof. See [8] page 64. □

Definition 2.5. The group H of Theorem 2.1 is the **cyclic subgroup** of G **generated** by P and it is denoted as $H = \langle P \rangle$.

Definition 2.6. If $\langle P \rangle = G$, P generates group G and P is a **generator** for G.

Definition 2.7. A group G is **cyclic** if some element P of G generates G.

Let $P \in G$, if the cyclic subgroup $\langle P \rangle$ is finite, the order of P is the order $\langle P \rangle$.

Theorem 2.2. Every cyclic group is abelian.

Proof. See [8] page 70. □

Theorem 2.3. A subgroup of a cyclic group is cyclic.

Proof. See [8] page 72. □

Discrete Logarithm Problem. Let $(G, \times)$ be a cyclic group of order n and let P be a generator of it. The Discrete Logarithm Problem (DLP) for G is to find the unique k such that;

$$Q = P^k = \overbrace{P \times P \times \ldots \times P}^{k \text{ times}} \tag{2.1}$$

where $Q \in G$ and $0 \leq k \leq n-1$.

Note that it becomes finding k such that $kP = Q$ if the group is defined additive. k exists since P is a generator of the group.

The complexity of finding k depends on the selection of the group and the binary operation. Discrete Logarithm Problem (DLP) is used as an underlying hard problem in many cryptographic protocols (i.e. Diffie Hellman Key Exchange Protocol).

Diffie Hellman Problem. For arbitrary k, m, r, Diffie Hellman Problem is closely related to DLP. Given P, P^k and P^m, finding P^{km} is called Diffie Hellman Problem (DHP). For certain groups it is assumed that the DHP is hard. Computing P^{km} is also referred as Computational Diffie Hellman Problem (CDHP) in order to separate it from Decisional Diffie Hellman Problem (DDHP) which is the problem of distinguishing between the distributions $\langle P; P^k; P^m; P^{km} \rangle$ and $\langle P; P^k; P^m; P^r \rangle$.

Definition 2.8. A set $\mathbb{F}$ together with two operations, which are addition (denoted by +) and multiplication (denoted by $\cdot$) is said to be a **field** if it satisfies the following properties;

- $(\mathbb{F},+)$ is an abelian group with additive identity element denoted by 0.
- $(\mathbb{F}\backslash\{0\},\cdot)$ is an abelian group with multiplicative identity element denoted by 1.
- $(P+Q)\cdot R = P\cdot R + Q\cdot R$ for all $P, Q, R \in \mathbb{F}$.

If $\mathbb{F}$ has finite number of elements then $\mathbb{F}$ is said to be a **finite field**.

Field Operations. A field is equipped with addition and multiplication. The other operations are done by means of these two operations. *Subtraction* is defined by means of addition as $P - Q = P + (-Q)$ where $P, Q \in \mathbb{F}$. $-Q$ is the **negative** of Q which is the unique element in $\mathbb{F}$ such that $Q + (-Q) = 0$. *Division* is defined by means of multiplication. Let $P, Q \in \mathbb{F}$ and $Q \neq 0$. $P/Q = P \cdot Q^{-1}$ where Q^{-1} is the **inverse** of Q which is a unique element in $\mathbb{F}$ such that $Q \cdot Q^{-1} = 1$.

Definition 2.9. There exists a finite field $\mathbb{F}$ of order q if and only if q is a prime power such that $q = p^n$ where p is a prime number and $n \in \mathbb{Z}^+$. If $n = 1$, $\mathbb{F}$ is called a **prime field** and denoted by $\mathbb{F}_p$. The prime p is the *modulus* of $\mathbb{F}_p$. If $n > 1$, the field $\mathbb{F}$ is said to be an **extension field**.

Example 2.4. In the following there are some examples of arithmetic operations in prime field $\mathbb{F}_{13}$.

- Addition: $12 + 3 = 2$, $(15 \; mod \; 13 = 2)$
- Multiplication: $5 \cdot 4 = 7$, $(20 \; mod \; 13 = 7)$
- Subtraction: $8 - 11 = 10$, $(-3 \; mod \; 13 = 10)$
- Inversion: $4^{-1} = 10$, $(4 \cdot 10 \; mod \; 13 = 1)$

2.1.2 Elliptic Curve over a Finite Field

The idea of using elliptic curves for cryptography was put forward independently by N. Koblitz [9] and V. S. Miller [10] in 1985. The security of elliptic curve based cryptographic systems depends on easy calculation of point multiplication on an elliptic curve and inability to compute multiplicand back knowing resulting point and the beginning point.

Simply, an elliptic curve is a cubic curve defined over two variables. Every elliptic curve can be represented in Weierstrass form (2.2). As it can be defined over real numbers, it can also be defined over a finite field.

Let p be a prime number and $\mathbb{F}_p$ a prime field, elliptic curve E over prime field $\mathbb{F}_p$, denoted by $E(\mathbb{F}_p)$, can be defined by the following equation;

$$y^2 = x^3 + ax + b \tag{2.2}$$

where $a, b \in \mathbb{F}_p$ should satisfy $4a^3 + 27b^2 \neq 0(\mod p)$. An elliptic curve can be used to form a group. Let $x, y \in \mathbb{F}_p$, a pair (x, y) is a point on the curve if it satisfies the equation. On an elliptic curve, there is also a special point called point at infinity which is denoted by ∞ or $\mathcal{O}$.

An elliptic curve group over $\mathbb{F}_p$ consists of the points on the corresponding elliptic curve, together with the point at infinity and a properly defined addition operation.

Informally, the addition operation is defined in the following way; suppose that P and Q are two distinct points on an elliptic curve. In order to add P and Q, a line is drawn through these two points. This line will intersect the elliptic curve in another point, $-T$. When we draw a perpendicular line to the x-axis we will find reflection of point $-T$ which is T. T is the resulting point as $T = P + Q$.

With this addition rule, the set of points $E(\mathbb{F}_p)$ forms an additive abelian group whose identity element is the point at infinity and each element P has its inverse $-P$ such that;

$$for\ P = (x, y)\ \text{ and }\ -P = (x, -y),$$

it is written,

$$P + (-P) = (-P) + P = \mathcal{O}.$$

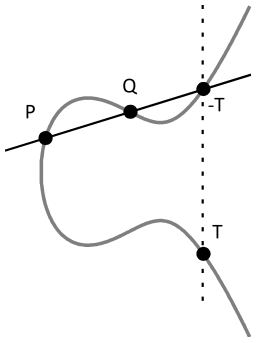

FIGURE 2.1: Addition on an elliptic curve

Elliptic Curve Discrete Logarithm Problem. Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$. Let P and Q be points in $E(\mathbb{F}_p)$. Finding an integer $m \in \mathbb{Z}^+$ such that;

$$Q = mP$$

is called the Elliptic Curve Discrete Logarithm Problem (ECDLP). The complexity of ECDLP depends on the selection of the curve. There are different algorithms to calculate the discrete logarithm. The fastest known algorithm to solve ECDLP in $E(\mathbb{F}_p)$ takes approximately $\sqrt{p}$ steps [11].

2.1.3 Cryptographic Pairings

Bilinear pairings are used in many cryptographic applications such as Boneh and Franklin's identity based encryption scheme [2]. The main idea of using pairings in cryptography is to construct a mapping between two groups which provides a reduction of one problem in one group to an easier problem in the other group. The most well known cryptographic pairings are the Weil pairing and the Tate pairing.

Let p be a prime, G_1 and G_2 be additive cyclic groups. Suppose that the order of G_1 and G_2 are p and G_3 is a multiplicative cyclic group of order p then a pairing is a map as :

$$e : G_1 \times G_2 \to G_3 \tag{2.3}$$

Note that G_1 and G_2 can also be the same groups.

Definition 2.10. Let G_1 and G_2 additive groups of order p with identity $\mathcal{O}$. Let G_3 is a multiplicative group of order p with identity 1. If the following properties are correct for all $P, P' \in G_1$ and all $Q, Q' \in G_2$ then e is said to be **bilinear pairing**:

- $e(P+P', Q) = e(P, Q) \cdot e(P', Q)$
- $e(P, Q+Q') = e(P, Q) \cdot e(P, Q')$
- $e(mP, Q) = e(P, Q)^m$

In addition to bilinearity, if there is an efficient algorithm to compute $e(g_1, g_2)$ for any $g_1 \in G_1$, $g_2 \in G_2$ and if e is non-degenerate which means the map does not send all pairs in $G_1 \times G_2$ to unity in G_3, then e is called **admissible bilinear pairing**. Admissible bilinear pairings are preferred in cryptographic systems because it is efficient to compute and hard to invert.

Pairings are also used over elliptic curves since they can be used to form groups as mentioned in 2.1.2. A pairing function maps a pair of points in one elliptic group to some elliptic curve group.

Definition 2.11. Let E be an elliptic curve defined over a field K_0 then the set

$$\mu_n = \{ u \in (\overline{K_0})^* : u^n = 1 \} \tag{2.4}$$

is called **n$^{\text{th}}$ root of unity** set.

$$(\overline{K_0})^* = K_0 - \{0\} \tag{2.5}$$

where K_0 is a field, K_0^* is the multiplicative group of K_0, $\overline{K_0}$ is the algebraic closure of K_0 and $(\overline{K_0})^*$ is the multiplicative group of $\overline{K_0}$.

Let $K = K_0(\mu_n)$ be the extension of K_0 generated by the n^{th} root of unity. n-torsion of $E(K)$ is defined as;

$$E(K)[n] = \{ P \in E(K) : [n]P = \mathcal{O} \}. \tag{2.6}$$

The point set of $E(K)$ is;

$$E(K) = \{(x,y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}. \tag{2.7}$$

Tate Pairing. Let $\mathbb{F}_q$ be a finite field with $q = p^n$ elements where p is prime and E be an elliptic curve defined over $\mathbb{F}_q$. The identity of E is denoted by $\mathcal{O}$.

Consider a large prime r such that $r \mid \#E(\mathbb{F}_q)$ and let k be the smallest integer such that $r \mid q^k - 1$ with $r^2 \nmid q^k - 1$.

If all conditions defined above are satisfied then

$$E[r] \subseteq E(\mathbb{F}_{q^k}) \tag{2.8}$$

where k is called **embedding degree** or **security multiplier**. When the embedding degree gets bigger, known algorithms for solving DLP in $\mathbb{F}_{q^k}$ gets slower.

Let f be a function on E then the intersection function of f and elliptic curve E can be defined as:

$$div(f_{r,p}) = r(P) - r(\mathcal{O}) \tag{2.9}$$

or

$$div(f_{r,p}) = s(P) - ([s]P) - (s-1)(\mathcal{O}) \tag{2.10}$$

where $f_{r,p}$ defines a function which is having s roots on point P and $div(f_{r,p})$ shows the points on f which intersect E.

Let $P \in E(\mathbb{F}_{q^k})[r]$ and $Q \in E(\mathbb{F}_{q^k})$ then Tate Pairing $t(.,.)$ is defined as;

$$t(P,Q) : E(\mathbb{F}_{q^k})[r] \times \frac{E(\mathbb{F}_{q^k})}{rE(\mathbb{F}_{q^k})} \to \mu_r \tag{2.11}$$

$$(P,Q) \mapsto f_{r,p}(Q)^{\frac{q^k-1}{r}} \tag{2.12}$$

where

$$rE(\mathbb{F}_{q^k}) = \{[n]P \quad : \quad P \in E(K)\} \tag{2.13}$$

The Tate pairing is well defined, bilinear, non-degenerate and efficiently computable. See Miller's algorithm [12] for computing the Tate pairing.

The **Weil pairing** [13] [12] is another bilinear pairing which is used in Boneh and Franklin's Identity Based Encryption scheme [2] which relies on Bilinear Diffie Hellman Problem.

Bilinear Diffie Hellman Problem. Let G_1, G_2 be two groups of prime order q. Let $e : G_1 \times G_2 \to G_3$ be an admissible bilinear map and let P be a generator of G_1. The Bilinear Diffie Hellman Problem (BDHP) in $\langle G1; G2; e\rangle$ is computing $W = e(P,P)^{kmr} \in G_2$ while $\langle P; P^k; P^m; P^r\rangle$ for some $k, m, r \in \mathbb{Z}_q^*$ are given.

DDHP in G_1 is easy and cannot be used to build cryptographic systems while CDHP can still be hard. The security of Boneh and Franklin's Identity Based system is based on BDHP which is a variant of CDHP. BDHP is hard in some certain groups, see [2].

2.1.4 Verifiable Secret Sharing

Secret sharing is a method of distributing a secret among multiple players as no player knows the whole secret but each player has a share for the secret. The secret can be reconstructed only if a sufficient number of shares are retrieved. In 1979 Shamir put forward a scheme called (n,k) threshold scheme [14]. The scheme is based on polynomial interpolation. Given k points in the form (x_i, y_i) where each x_i is distinct , there is only one polynomial $f(x)$ of degree $k-1$ such that $f(x_i) = y_i$ for all i.

Verifiable secret sharing was first introduced by Chor, Goldwasser, Micali, and Awerbuch in 1985 [15]. In such a scheme each player can verify its share. Feldman's verifiable secret sharing scheme [16] is based on Shamir's scheme and it improves Shamir's scheme by including verifiability. By using a verifiable secret sharing scheme any dishonest or malicious node in the system can be found.

2.2 Identity Based Cryptography

In an identity 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 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.

In 2001, Boneh and Franklin devised the first practical implementation of such an Identity-Based Encryption scheme [2]. Their approach uses bilinear maps and relies on the BDH Assumption.

2.2.1 Encryption

Encryption using identity based cryptography works as in Figure 2.2. Node A wants to send an encrypted message to node B. In order to encrypt the message M, instead of asking for B's public key it uses the identity string which defines B. H is a hash function, which is known by each node in the system and used for creating public keys of equal length and same form from the id strings. It is important to note that, A can perform the encryption even if B doesn't have its private key yet. After receiving the cipher-text C, B can use its private key to decrypt the message if it already has its private key. Otherwise, B asks to the private key generator (PKG) for its private key. PKG generates the private key d_{ID_B} for node B using hashed value of B's identity string $H(ID_B)$ and the master private key s.

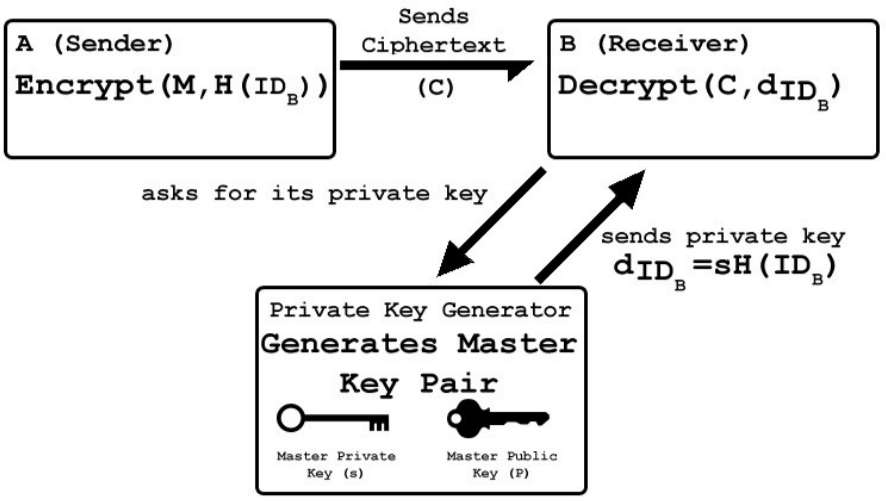

FIGURE 2.2: Identity based encryption scheme

2.2.2 Signature

Figure 2.3 shows how the identity based signature scheme works. When node *A* wants to send a signed message to node *B*, it uses its own private key to sign the message. It is worth to note that, if it doesn't have, node *A* should ask private key generator for its private key to be able to sign a message. Private key generation is the same as it is in the encryption - decryption scheme.

After receiving the signed message *SM*, node *B* can use the hashed value of *A*'s identity string to verify the signature. Verification of the signature doesn't require any certification authority or other trusted third parties.

Private key generation is only performed in such cases like the establishment of the system, adding a new node to the system, when a renewal is needed, etc.

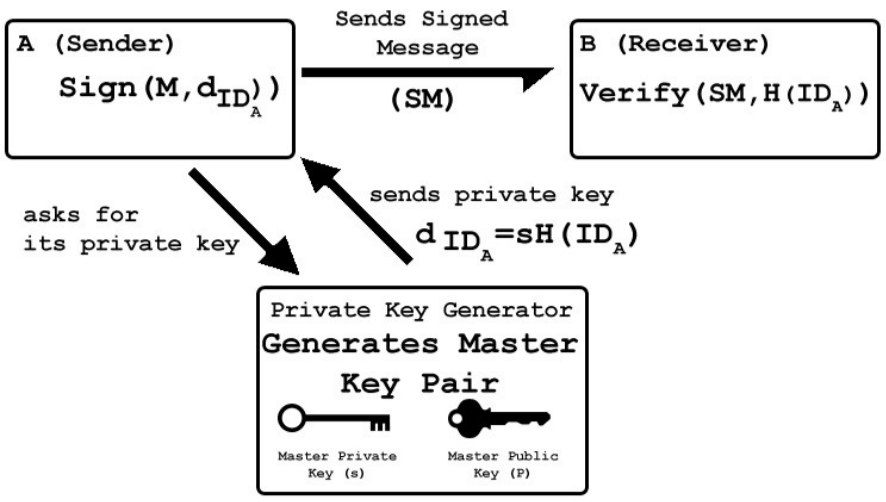

FIGURE 2.3: Identity based signature scheme

In section 3.2 an implementation of identity based signature on industrial control systems is presented. The possibility to provide the low level devices of Supervisory Control And Data Acquisition (SCADA) infrastructures with a mechanism allowing to obtain automatically a private key for signing their own network packets while limiting at the minimum interactions with the external world to verify a signed packet, is extremely appealing. Since each entity in a SCADA system already has a unique identity string known by everyone in the system (e.g. the PLC *id* for the low

level components and, eventually the MAC address for the SCADA servers), identity based cryptography is a very useful and suitable scheme to provide authenticity, integrity and confidentiality for SCADA systems.

2.2.3 Key Escrow Problem

Although an identity based cryptographic system does not require a certification authority, there is a need for a PKG. Basically, a trusted third party signs user's publicly known string using the master private key in order to generate a private key for the user. Since the PKG can generate private keys for users, it can sign or decrypt messages of any user or it can make users' private keys public. This problem about private key generation is called *key escrow problem*. Distributed private key generation (DPKG) is one of the solutions to the key escrow problem. In both schemes described in [2] and [17], secret sharing methods are used for distributing private key generation among multiple PKGs.

2.2.4 Distributed Private Key Generation

As it is explained in section 2.2.3 the problem to be solved about identity based scheme is the key escrow problem. In this thesis, in order to eliminate the problem, a completely distributed private key generation scheme is performed. In this scheme, a number of PKG nodes participate while they share the responsibility equally. The private key generation scheme consists of three steps: setup, distribution and extraction. A bulletin board, n PKG nodes and clients who need to have private key are the players in the distributed private key generation process. Algorithms presented here are based on schemes described in [17], [2] and [16].

2.2.5 Setup

This step is a preparation step to create the system parameters and to get ready for creating the master key pair distributively and extraction of private keys. It is performed by the bulletin board application. The algorithm for setup phase of DPKG is displayed in the algorithm 1.

To perform the setup for DPKG, given K as the security parameter, a prime q of size K is selected. An elliptic curve group $\mathbb{G}$ of order q and a cyclic multiplicative group

Algorithm 1 Bulletin Board setup

Require: security parameter K and cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}^*$

$q \leftarrow$ generate prime of size K
$\mathbb{G} \leftarrow$ choose elliptic curve group of order q
$\mathbb{G}_T \leftarrow$ choose cyclic multiplicative group of order q
$g \leftarrow$ find a generator of $\mathbb{G}$
choose pairing function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
$C[n][k] \leftarrow createcommitments()$
for $i = 0 \rightarrow n$ **do**
    for $j = 0 \rightarrow k$ **do**
        $C[i][j] \leftarrow 0$
    end for
end for
$C_f[k] \leftarrow createfinalcommitment()$
for $i = 0 \rightarrow k$ **do**
    $C_f[i] \leftarrow 0$
end for
while system is on **do**
    broadcast(q, $\mathbb{G}$, $\mathbb{G}_T$, e, H, g)
end while

of order q $\mathbb{G}_T$ are also selected. A cryptographic hash function has to be chosen as $H : \{0,1\}^* \rightarrow \mathbb{G}^*$. Then a generator g of group $\mathbb{G}$ is chosen. It is important to note that any element of group $\mathbb{G}$ except the identity element can be a generator since $\mathbb{G}$ is a cyclic additive group of a prime order. A bilinear pairing function $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is chosen to be used in the verification part of the key extraction. Commitment vectors for PKG nodes, $C[i][j]$ and the final commitment vector $C_f[j]$ are created and initialized by setting to zero, where $i = 1, 2, \ldots, n$ and $j = 0, 1, \ldots, k-1$. All above public system parameters are broadcasted by the bulletin board.

2.2.6 Distribution

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. To do this, we use an improved version of (n,k) Feldman's secret sharing scheme which is stated in [17].

Since the private key generation for a client means to sign client's hashed ID with the master private key, no one is allowed to have access the master private key. Thus, we distribute master private key among PKG nodes in a way nobody knows it by using secret sharing approach. 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.

The following steps are performed by every node PKG_i where $i = 1, 2, \ldots, n$.

Algorithm 2 PKGi distribution

Require: i

Require: system parameters (q, $\mathbb{G}$, $\mathbb{G}_T$, e, H, g) are broadcasted on Bulletin Board

1: //create a polynomial $f_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + \ldots + a_{i(k-1)}x^{(k-1)}$
2: **for** $j = 0 \rightarrow k-1$ **do**
3: $f_i[j] \leftarrow$ random element( )
4: **end for**
5: //calculate commitments to be broadcasted on Bulletin Board
6: **for** $j = 0 \rightarrow k-1$ **do**
7: $C_i[j] \leftarrow f_i[j]g$
8: **end for**
9: send C_i to Bulletin Board
10: **for** $j = 1 \rightarrow n$ **do**
11: subshare[j] $\leftarrow$ evaluate (f_i, j)
12: **end for**
13: **for** $j = 1 \rightarrow n$ **do**
14: **if** $j \neq i$ **then**
15: send(PKGj, subshare[j])
16: **end if**
17: **end for**
18: **while** not receive all **do**
19: wait for subshares from other PKG nodes
20: **end while**
21: subshares from n-1 PKG nodes are received
22: //subshares[k] is the set of subshares received
23: //num[k] is the set of PKG numbers. i.e. subshare[3] is sent by PKG num[3]
24: $s_i \leftarrow 0$
25: **for** $j = 1 \rightarrow n$ **do**
26: $s_i \leftarrow$ add(s_i, $subshare_j$)
27: //where $subshare_j$ was calculated by PKGj for PKGi
28: **end for**
29: check1 $\leftarrow s_i g$
30: check2 $\leftarrow 0$
31: **for** $j = 0 \rightarrow k-1$ **do**
32: check2 $\leftarrow add(check2, i^j C_f[j])$
33: **end for**

```
34: if check1 = check2 then
35:     //share s_i is correct
36: else
37:     for m = 0 → n do
38:         check1 ← subshare[m]g
39:         check2 ← 0
40:         for l = 0 → k − 1 do
41:             check2 ← add(check2, i^l C[num[k][l]]
42:         end for
43:         if check1 = check2 then
44:             //share subshare[m] is correct
45:         else
46:             //PKG num[m] sent incorrect subshare
47:             //ask PKG num[m] again
48:             //or mark PKG num[m] as malicious exclude it and repeat distribution
49:         end if
50:     end for
51: end if
```

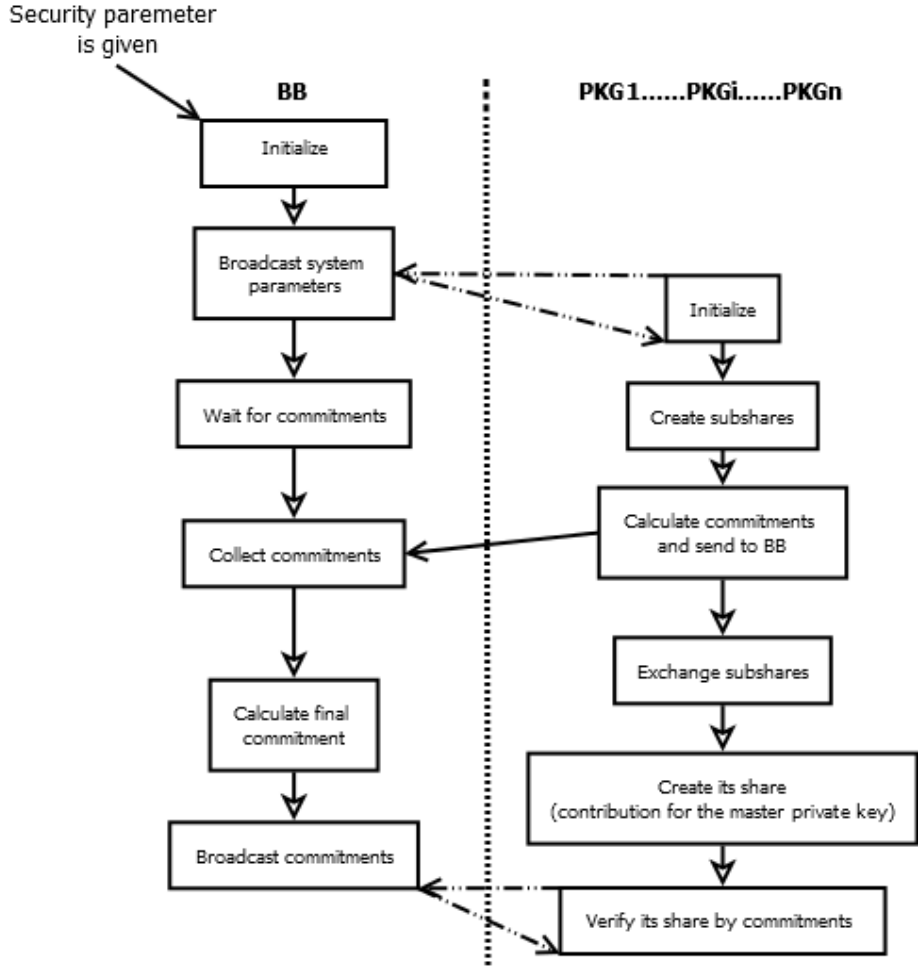

FIGURE 2.4: Block schema of the distribution step of private key generation.

To perform the distribution step of DPKG, first of all a random polynomial is created by PKG_i as:

$$f_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + \ldots + a_{i(k-1)}x^{(k-1)} \tag{2.14}$$

where $f_i(x) \in \mathbb{Z}_q[x]$ and the degree of the polynomial is $k-1$. a_{i0} is PKG_i's contribution to the master private key and $a_{i0}g$ is the contribution to the master public key. Node PKG_i computes subshares by evaluating Function (2.14) for $j = 1, 2, \ldots, n$. It then sends $s_{ij} = f_i(j) \bmod q$ to node PKG_j over a secure channel. It also sends commitment vector $C_{il} = a_{il}g$ for $l = 0, 1, \ldots, k-1$ to be broad casted. Final commitment values C_f are updated and broad casted for as $C_f[l] \leftarrow C_f[l] + C[i][l]$ $l = 0, 1, \ldots, k-1$. After each node receives $n-1$ subshares from other nodes, it calculates its secret share as $s_j = \sum_{j=1}^{n} s_{ij}$. Then it verifies its secret share by checking;

$$s_j g \stackrel{?}{=} \sum_{l=0}^{k-1} j^l C_f l \tag{2.15}$$

If the secret share is not verified, node PKG_j checks the subshares it received from each node PKG_i by using Formula (2.16) in order to find the dishonest node.

$$s_{ij} g \stackrel{?}{=} \sum_{l=0}^{k-1} j^l C_{il} \tag{2.16}$$

Bulletin Board collects commitments from all n PKG nodes then it updates final commitment C_f for each PKGi as $C_f[j] \leftarrow C_f[j] + C[i][j]$ $j = 0, 1, \ldots, k-1$. After all the commitments are collected and final commitment is calculated, Bulletin Board broadcasts all commitments.

Note that, the master secret key is $s = \sum_{i=1}^{n} a_{i0}$ but none of the PKG nodes knows it since each of them has only its own secret, a_{i0}. Master public key is $Pub = \sum_{i=1}^{n} a_{i0}g = C_f[0]$. Public key share for the node PKG_j is $Pub_j = \sum_{l=0}^{k-1} j^l C_f[l]$. C_f, C_{il}, Pub and Pub_i must be known by all the PKG nodes, where $i = 1, 2, \ldots, n$ and $l = 0, 1, \ldots, k-1$. These are broad casted by bulletin board application.

Keeping the threshold number (k) constant while incrementing the number of nodes increases the reliability of the system as well. Indeed a client has a wide range of available PKG nodes to choose the k needed to compute its private key.

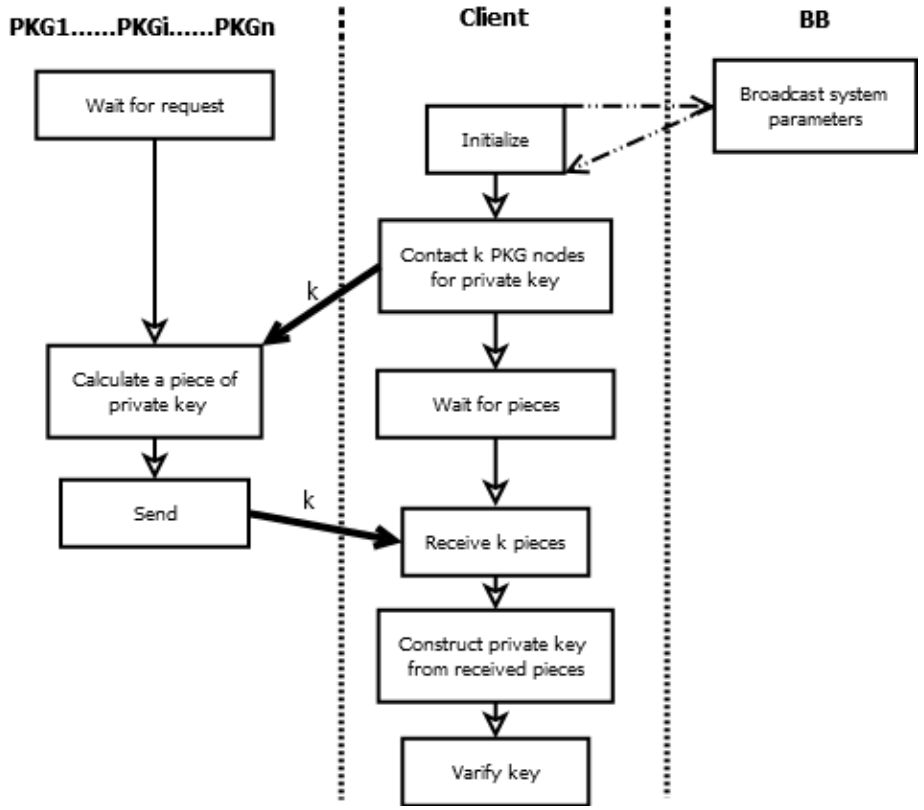

FIGURE 2.5: Block schema of the extraction step.

2.2.7 Extraction

Let $\mathcal{O}$ be the set of available PKG nodes and g, e, H, Pub, Pub_i are known by both PKG nodes and clients where $i = 1, 2, \ldots, n$.

Client A with identity ID_A contacts k alive nodes from $\mathcal{O}$. Each node PKG_i returns a private key piece as $s_i \text{H}(ID_A)$ over a secure and authenticated channel. After receiving k pieces from k available PKG nodes, client A constructs its private key by following Algorithm 3.

After receiving k pieces, client A constructs its private key by evaluating the formula;

$$d_{ID_A} = \sum_{PKG_i \in \mathcal{O}} \lambda_i s_i H(ID_A) \tag{2.17}$$

where the Lagrange coefficient is;

$$\lambda_i = \prod_{PKG_j \in \mathcal{O} \setminus \{i\}} \frac{j}{j - i} \tag{2.18}$$

Algorithm 3 Client extraction

Require: $\mathcal{O}$, k pieces, $g, e, H, Pub[n]$ to be known, where $i = 1, 2, \ldots, n$ and master public key Pub.

Require: PKG numbers are known

```
1: //s[k] is the set of pieces received
2: //num[k] is the set of PKG numbers which sent a piece to the client
3: //s[1] is the piece received from PKG num[1]
4: d ← ∑_{PKG_i ∈ O} λ_i s_i H(ID_A)
5: // where λ_i = ∏_{PKG_j ∈ O\{i}} j/(j−i)
6: check1 ← pairing_e(d, g)
7: check2 ← pairing_e(H(ID), Pub)
8: if check1 = check2 then
9:    //the key is verified
10: else
11:    //key is not verified find the malicious node
12:    m ← 0
13:    for j = 0 → k − 1 do
14:       check1 ← pairing_e(s[j]H(ID), g)
15:       check2 ← pairing_e(H(ID), Pub[num[j]])
16:       if check1 ≠ check2 then
17:          //PKG num[j] sent malicious piece
18:          m ← num[j]
19:       end if
20:    end for
21:    if m ≠ 0 then
22:       //ask PKG node m to send the piece again and repeat calculation of d
23:       // or exclude PKG node m and start again
24:    else
25:       // repeat extraction starting from asking to k PKG nodes for pieces
26:    end if
27: end if
```

The client A can verify the its private key by checking;

$$e(d_{ID_A}, g) \stackrel{?}{=} e(H(ID_A), Pub) \tag{2.19}$$

In the algorithm e is a bilinear pairing function. As a result of bilinearity property of the pairing function, the dishonest node can be recognized by observing an inequality in the pairing checks.

$$e(d_{ID_A}, g) = e(H(ID_A), Pub)$$

$$e(sH(ID_A), g) = e(H(ID_A), sg)$$

If the verification is unsuccessful, it then can verify each received piece $s_i H(ID_A)$ by checking;

$$e(s_i H(ID_A), g) \stackrel{?}{=} e(H(ID_A), Pub_i) \tag{2.20}$$

The dishonest node PKG_i can be recognized by observing an inequality in Formula (2.20).

Whenever a new node is included to the system it performs this extraction part to have a private key so that it can start communicating with the other system nodes. However when the system requires a change of the master key, all three steps (setup, distribution, extraction) have to be executed from the beginning.

Chapter 3

An Implementation and Practical Analysis of Distributed Private Key Generation for Identity Based Cryptography

3.1 Distributed Private Key Generation Implementation

Distributed private key generation (DPKG) is needed to generate keys for Identity Based Cryptography however it is used in some other cryptographic systems as well. Identity based cryptography is advantageous when the public key management is difficult for the system. In our work we have used the DPKG implementation for securing Supervisory Control and Data Acquisition (SCADA) systems since it is highly appropriate area due to its security need and its shortcomings that makes the traditional key management methods not feasible to implement. In this section after a short look at the implementation, a case study on SCADA systems is presented.

3.2 Case Study: Implementing DPKG on Industrial Control Systems

This part of the study has been done in European Commission Joint Research Centre, Ispra, in collaboration with Igor Nai Fovino and Marco Taddeo.

In this digital age, one of the most important problems is the security threats related to the Information and Communication Technologies (ICT). All systems deploying ICT are prone to failures and vulnerabilities that can be exploited by malicious software and agents. Modern Critical Infrastructures (e.g. Power Plants, Water Grids etc.) largely use ICT in order to provide new services and offer new features. In particular, according to a relatively new trend, several of the maintenance and management operations related to such installations are conducted remotely taking advantage of public networks (i.e. the Internet). Using these facilities contributes to make many operations taking place in critical infrastructures faster and easier. It also reduces maintenance costs (less operators, unified control centers etc.) and improves efficiency. Despite the advantages, ICT and Internet expose the critical infrastructures to new range of security threats. Since the security of the critical infrastructures have great importance and effect on the security of citizens, the problem described above requires urgent solution.

The use of traditional ICT protection techniques (Firewalls, Intrusion Detection Systems, Anti-viruses etc.) for securing process networks only mitigates the risk of successful attacks [18], mainly limiting those caused by traditional issues in the office environment. In fact, traditional firewalls and anti-virus can easily, if well configured, stop known ICT attacks, for which they have the proper signatures and detection rules. However those networks remain exposed to other threats (for more details please refer to the subsection 3.2.1). The motivation of such level of exposure is mainly connected to the intrinsic weakness of the communication protocols used in the “Supervisory Control And Data Acquisition” (SCADA) systems to monitor and control the field devices.

Several SCADA systems are geographically very sparse (e.g., a gas pipeline) and composed by thousand of elements, making the key management process extremely complex and resource consuming. The use of certification authorities and the use of centralized repositories for public key management etc. result generally in a too heavy and complex burden to be considered a valid solution for distributed SCADA systems. To solve this problem, we use a key management scheme that deploys

an identity-based approach. Identity based cryptographic schemes are commonly based on using a rather sophisticated mathematical concept, e.g. pairings defined over elliptic curves.

3.2.1 Process Control System Vulnerabilities

Process Control Systems (PCS) are used in industrial installations to control the production process. In other words, they are used to monitor and manage the field devices (valves, pumps heaters etc.) in charge of performing operations on the field. PCSs are normally used to control installations such as power plants, nuclear plants, chemical installations, gas pipelines etc. It is evident that how their security should be considered critical. In the following we firstly provide brief description of the relevant bricks of a PCS, then we present the related vulnerability problems and finally we discuss the desirable features that a possible countermeasure should implement.

3.2.1.1 PCS Architecture

The core of PCSs is normally the Supervisory Control and Data Acquisition system (SCADA). The basic components characterizing a SCADA system are:

- **Master Terminal Unit (MTU)**: The MTU presents data to the operator, gathers data form the remote Programmable Logic Controllers (PLCs) and actuators site, and transmits control signals. It contains the high level logic of the industrial system under control.
- **Remote Terminal Unit (RTU)**: It acts as a slave in the master/slave architecture.It sends control signals to the device under control, acquires data from these devices, receives commands from the MTU and transmits the data gathered to the MTU. An RTU may be a PLC.
- **Field Devices**: They are the physical devices in charge of executing the operations requested by the PLCs and the MTU.

Figure 3.1 shows how a SCADA system is integrated in a typical turbo-gas power plant facility: SCADA servers are located into a *process network*, while the RTUs (or the PLCs) are located into the *field network*. Between these networks exists

the *control network*. The masters are also controlled remotely by systems into the company intranet, or through a remote connection using typically a site to site VPN through the public network.

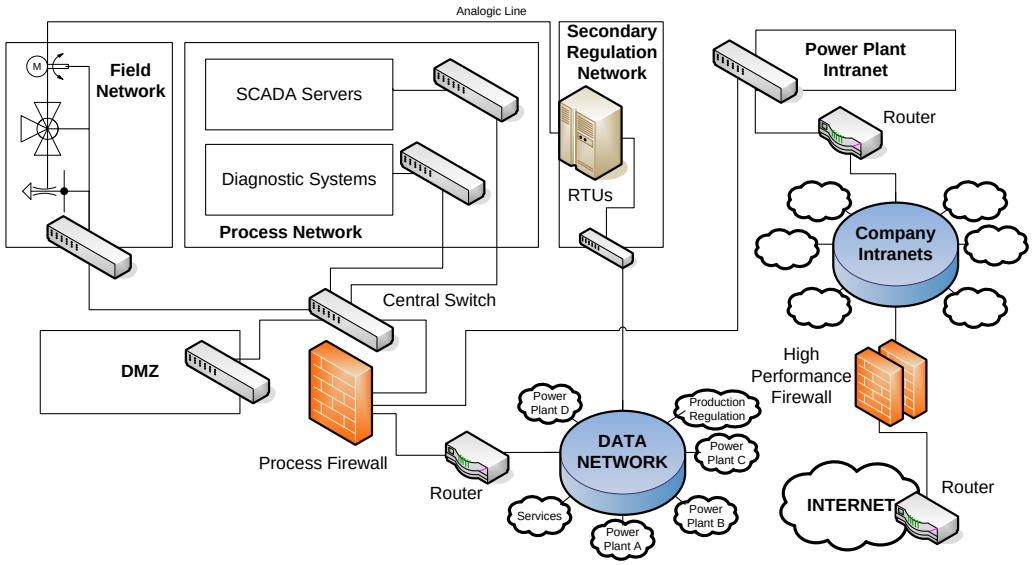

FIGURE 3.1: A typical SCADA system embedded into a turbogas power plant facility

The core of the control network of every SCADA system is the communication protocol (e.g. Modbus, Profibus, DNP3 etc.) used to collect data and dispatch commands. In this study we focus our attention on the Modbus protocol but the architecture proposed can be easily applied to all the other known SCADA protocols. Modbus is an application layer messaging protocol, positioned at level 7 of the OSI model (in the case of Modbus over TCP), which provides client/server communication between devices connected on different types of buses or networks. Communications can be (i) query/response type (communication between a master and a slave), or (ii) broadcast response type where the master sends a command to all the slaves. A transaction comprises one single query and single response frame, or one single broadcast frame. A Modbus frame message contains the address of the intended receiver (e.g. PLC number), the command to be executed by the receiver and eventually the data needed for the execution of such a command. Modbus/TCP embeds a Modbus frame into a TCP frame [19]. All the functions supported by the Modbus protocol are identified by an index number (e.g. *01 (0x01) Read Coils* is a function used to read the status of the coils in a remote device). These functions can be *operational* (write/read registers, coils etc.) or *diagnostic*. The Modbus communication interface was originally designed for a multi-drop network based

on a master/client serial architecture. Due to the fact that, when designed, the deployment was in isolated, tightly controlled systems. All security aspects such as *integrity, authentication, non-repudiation* etc. were not taken into consideration.

3.2.1.2 Modbus Vulnerabilities

The porting of Modbus over TCP/IP has obviously introduced new layers of complexity in the management of the reliable delivery of control packets in an environment with strong real time constraints. In addition, it has opened new possibilities to attackers motivated to cause damages to target industrial systems. In particular, this protocol (and generally all the SCADA protocols):

1. Do not apply any mechanism for checking the integrity of the command packages sent by a master to a slave and vice-versa.
2. Do not perform any authentication mechanism between master and slaves, i.e. everyone could claim to be the "master" and send commands to the slaves.
3. Do not apply any non-repudiation and anti anti-replay mechanisms.

These security shortcomings can be used by malicious users for performing different kind of attacks:

- **Unauthorized Command Execution:** The lack of authentication between master and slave can be used by attackers to forge packets which can be directly sent to a pool of slaves. This is, specifically considering the use of SCADA system in critical infrastructures like Nuclear plants, and extremely dangerous class of threats, allowing to a well determined attacker to change the entire configuration of a field network.
- **SCADA-DOS:** On the basis of the same principle, an attacker can also forge meaningless Modbus packets, always impersonating the master, and consume the resources of the RTU
- **Man-in-the-Middle attacks:** The lack of integrity checks allows attackers to access the production network for implementing typical Man-in-the-Middle (MITM) attacks, modifying the licit packets sent by the master.

- **Replay-Attacks:** The lack of anti-replay mechanisms allows attackers to reuse captured legitimate Modbus/DNP3 packets.

Huitsing et all [20] presented a wide overview of the Modbus vulnerabilities; moreover Nai et all [21] presented a first proof of concept of a ad-hoc created malware able to take the control of an entire power plant field network (i.e. able to directly send commands to the actuators of the power plant), by taking advantage of the vulnerabilities of Modbus.

3.2.1.3 Distributed Identity Based Secure Modbus

As it is possible to guess from the previous section, several of the vulnerabilities related to the PCS are due to the weaknesses of the industrial communication protocols used to control the field devices. An apparently quick solution to the problem would be to introduce in the communication channel in between master and slaves a cryptographic layer allowing to guarantee the traditional properties of:

- *Authentication*
- *Confidentiality*
- *Integrity*

Among those requirements, the *confidentiality* can be considered as an unnecessary issue for the security of SCADA, since the knowledge of the fact that a certain command sent from a master to a slave is not considered a sensible information. The most important and urgent requirement is authentication to prevent unauthorized command executions since it can cause severe damage on a critical infrastructure in case an attacker finds a way to sneak to the system and sends the SCADA devices malicious commands.

A signature implementation can be used for providing the system with authentication but the public key management is a problem for a SCADA system considering the low power computation of the field devices, low memory space available and limited network bandwidth. In addition, several of these systems are very sparse and composed by thousand of elements, making the key management process a complex problem.

The traditional way of public key management is using Certification Authorities (CA) but using a CA requires to open the critical infrastructure to external world which makes the infrastructure open to external attacks. In this point building a private Certification Authority for the system can be considered. However, it is still unlikely to use CA for the following reasons:

- PLCs are not capable of handling certificates
- For each message public key verification is needed
- PLCs do not have enough memory for local repositories
- Open to single point of failure

Instead of CA, using identity based approach is more advantageous for a SCADA system considering the below facts:

- Each element in a SCADA system has an identity string (i.e. PLC number)
- SCADA communication protocols already use these id strings in messages
- Verification is done only during system establishment and when master key is needed a renewal
- Distributed key generation eliminates the key escrow problem and single point of failure risk

In the following, we present the first working prototype of a secure Modbus protocol based on identity based signature with distributed private key generation.

3.2.2 Prototype Overview

Our Distributed PKG is implemented as sequential steps where different parts communicate each other by the TCP/IP protocol in a Linux environment. In this environment, we have a bulletin board, n PKG nodes and numerous clients as system entities. The prototype is implemented in C for two main reason: The first is that the Pairing Based Cryptography (PBC) library [22] is used for the underlying low level elliptic curve and finite field operations and the second reason is the low level control of the information exchanged by the nodes.

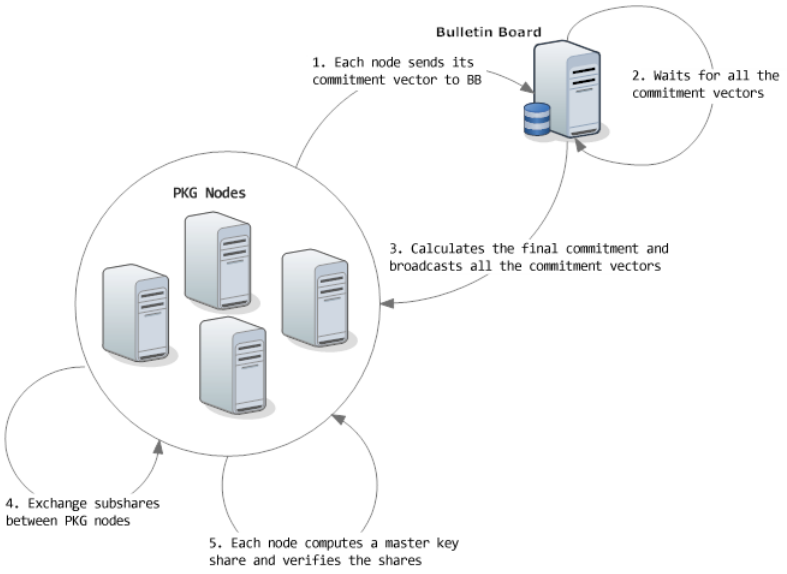

FIGURE 3.2: Distribution step

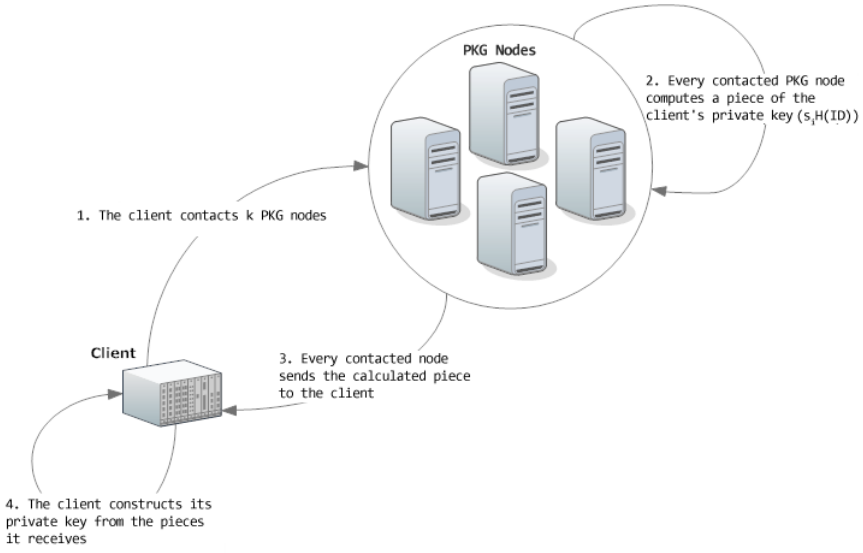

FIGURE 3.3: Key extraction step

In the start-up phase, the bulletin board loads the parameters and generates randomly the curve generator. This value is public and is stored in a file shared by the entire system and will never change once initiated. A list containing the network topology information is shared between all the PKG nodes available in the system. In particular, each node has a special file with a name, an IP and a listening port of the system's PKG.

No client can contact the nodes until the build-up phase is completed and master-key is created and shared because the system is closed and there are no sockets opened

to the outside.

Figure 3.2 and Figure 3.3 demonstrates the prototype we have developed.

3.2.3 Experimental Tests

To verify the efficiency and the validity of our approach, several comparative tests were performed on a protected environment which reproduces the network of a typical Gas Power Plant. The test was organized in three steps:

- In the distribution step, the distribution phase performance were tested.
- In the extraction step, we focused our attention on the client key extraction.
- In the signcryption step, we performed some tests on signing Modbus packages.

All the computers that the tests ran on were Linux based (Ubuntu 10.10).

3.2.3.1 Distribution

In order to measure the time required for completing the distribution phase with different numbers of PKG nodes, we performed several tests by keeping all other system parameters constant. More specifically, we changed the number of nodes (n) from 6 to 18 while keeping the threshold number (k) and other system parameters constant in order to see how the number of nodes effects the start-up time (time for completing the setup and the distribution steps).

# of PKG nodes	Avg. Time
2	162
3	225
4	283
5	313
6	329
7	341
8	350

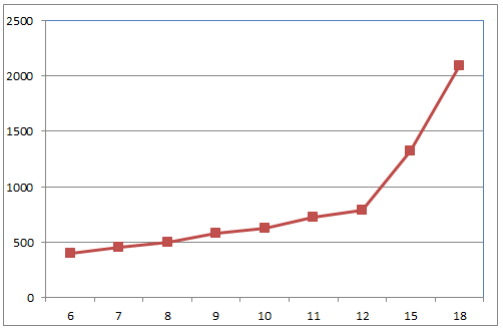

FIGURE 3.4: Distribution phase performance test (Time in ms.)

Graph displayed in Figure 3.4 shows how the time increases exponentially with the number of nodes (n). PKGs exchange $n \times n$ messages during setup and distribution steps. Considering that this setup phase needs to be executed only during system establishment or whenever a master key renewal is needed, an average time of 2 seconds is considered acceptable.

3.2.3.2 Extraction

In this test phase we focused on the time required by the client for extracting its private key. We changed the threshold number (k) from 2 to 8 while keeping the number of nodes (n) and the other system parameters constant. Changing the threshold value, the number of PKG nodes that the client needs to contact for key extraction changes. The value displayed in Figure 3.5 indicates the total extraction time together with the network delay and the .

# of PKG nodes	Avg. Time
2	162
3	225
4	283
5	313
6	329
7	341
8	350

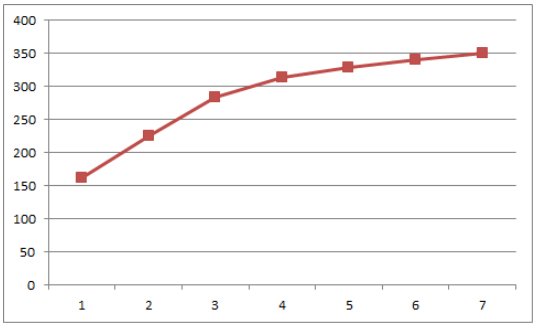

FIGURE 3.5: Key extraction phase performance test (Time in ms.)

This test shows that the performance of the key extraction phase mostly depends on the network delay. The time increases logarithmically with the threshold value. In fact the client has to contact k PKG nodes and every signed piece $(s_i H(ID))$ retrieval increases the total time. Keeping the threshold value too low will cause the system to be more exposed to an attacker trying to capture the user's private key, while keeping it too high will slow down the key extraction process. When the threshold value is higher than $k = \lfloor n/2 \rfloor + 1$ we also sacrifice reliability considering the case that some of the PKG nodes crashes.

By using a (n,k) threshold scheme for secret sharing which is summarized in Section 2.1.4 with $n = 2k-1$ we get a robust key management scheme. Even if $\lfloor n/2 \rfloor = k-1$ of the n pieces are destroyed, we still can recover the original key. On the other hand, our opponents cannot reconstruct the key even if they retrieve $\lfloor n/2 \rfloor = k-1$ of the remaining k pieces.

According to the practical analysis performed, it can be said that the selection of n between 7 and 14 and k between 4 and 7 (complying with the equation $n = 2k - 1$) is acceptable and applicable in terms of security, reliability and speed.

Client	Time
2	346
3	338
8	352
16	418
32	461

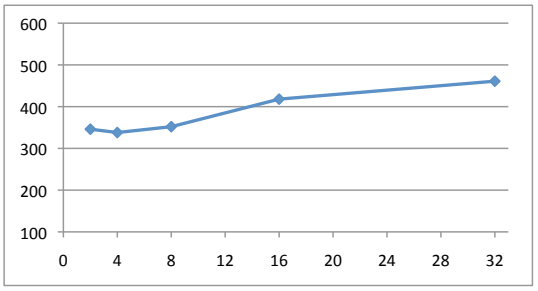

FIGURE 3.6: Simultaneous answer performance test (Time in ms.)

In most environments, many PLCs and masters can be turned on or connected to the system simultaneously. In order to verify the performance of the DPKG system we performed several test increasing the number of clients that simultaneously communicate with the DPKG system. Different numbers of clients contact the DPKG system to extract their private keys. The test is performed contacting 8 PKG nodes and number of clients that simultaneously contact to the PKG nodes rages from 2 to 32. The results displayed in Figure 3.2.3.2 show that there is not an explicit limit to the simultaneous requests. The implicit limits are only due to the computational resources of the machines.

3.2.3.3 Signature & Verification

The goal of this test is to present some preliminary values of a signature implementation between two SCADA entities: master and slave. The communication is performed through the Modbus protocol [23] [19]. In particular, a master signs a Modbus package by using the private key generated in the previous step and sends it to a slave (PLC). The client verifies the data integrity and sender's identity by following the identity based signature scheme described in Section 2.2.

Throughout the testing phase we used type A elliptic curves. This type of curve is described by the equation

$$y^2 = x^3 + x. \tag{3.1}$$

Read Coils	r=150 q=160	r=80 q=224	r=160 q=512
Req-sign	36.51	49.10	286.64
Req-ver	15.91	15.75	202.57
Resp-sign	31.71	47.07	281.25
Resp-ver	20.34	35.26	211.78

(a) Read coils command

WriteCoils	r=150 q=160	r=80 q=224	r=160 q=512
Req - sign	35.64	49.30	281.93
Req - ver	15.75	15.80	202.03
Resp - sign	30.65	49.13	280.00
Resp - ver	17.67	16.95	200.50

(b) Write coils command

FIGURE 3.7: Time spent for signing and verifying a Modbus package

The curve is defined over the field $\mathbb{F}_q$ for some prime $q = 3 \mod 4$. The order r is some prime factor of $q+1$. Moreover, we used a symmetric pairing where $\mathbb{G}_1$ and $\mathbb{G}_2$ are the set of points $E(\mathbb{F}_p)$.

The test is performed in two parts as:

- Master sends a ReadCoilsRequest to slave and waits ReadCoilsResponse;
- Master sends a WriteCoilsRequest to slave and waits WriteCoilsResponse.

Tables below show the total time spent (in msec) for signing and verifying Modbus packages according to the length (in bits) of selected security parameters r and q where $q+1 = r*h$.

As it is possible to see in Figure 3.7(a) and 3.7(b) the total time spent rages from minimum 17 msec to maximum 380 msec. Considering that we used a non-optimized prototype, we can consider that these results are quite promising.

3.2.4 Conclusion for the Case Study

SCADA systems are usually used to control critical infrastructures which makes security of them highly essential. The most important vulnerability of SCADA systems is the lack of authentication mechanism due to the SCADA protocols. In this study after proposing the use of an identity based signature scheme to secure SCADA protocols, we presented a completely distributed scheme for generating identity based private keys for the entities involved in a SCADA system. The advantages of this approach are not negligible. The use of identity based signatures makes the use of certificates and certification authorities unnecessary as well as it

makes the use of centralized repositories for storing public keys also unnecessary. It provides SCADA systems with authenticity efficiently. Moreover, after performing a private key generation the keys can also be used for encryption to provide confidentiality if it is needed. The presented distributed architecture provides robustness against ICT attacks and finds its perfect application in those geographically sparse SCADA systems composed of a huge number of active elements. The tests presented show how this approach is applicable for the SCADA environment in terms of security, reliability and time.

Chapter 4

Using Petri Nets for Modeling and Analyzing Cryptographic Protocols

4.1 Introduction to Petri Nets

The mathematical concept to describe systems with distributed states was developed by Carl Adam Petri in 1962 in his thesis [24]. With his dissertation he laid the basis of Petri net theory.

Petri net is a mathematical and graphical modeling tool useful for modeling concurrent, distributed, asynchronous, parallel, nondeterministic, and/or stochastic systems. As a graphical tool, it can be used as a visual-communication aid similar to flow charts, block diagrams and networks. Tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, etc.

The concept of Petri nets is a result of searching methods for describing and analyzing information and control data flow in information processing. The Petri net theory is well suited to describe the dynamic behavior of complex concurrent systems based on graph theoretical concepts.

In this chapter a brief summary of Petri net theory is given with respect to the notation and terminology of [25] and [26].

Definition 4.1. Formally, a net is a bipartite, non-empty, directed graph without isolated nodes and it is represented as;

$$N = (B, E, F)$$

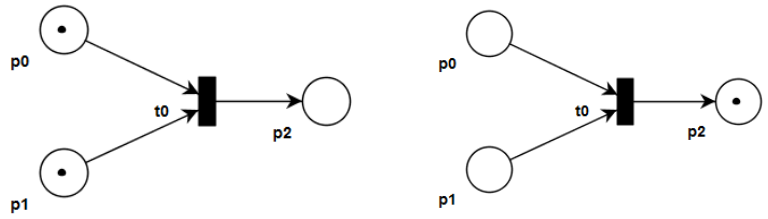

FIGURE 4.1: Demonstration of firing rule: (a) t_0 is enabled (b) t_0 is fired

where

- $B = b_1, b_2, \ldots b_m$ is a finite set of places,
- $E = e_1, e_2, \ldots, e_n$ is a finite set of transitions,
- $F \subseteq (B \times E) \bigcup (E \times B) \in$ is a set of arcs,
- $B \cap E = \varnothing$ and $B \cup E \neq \varnothing$.

Set B is also considered as the set of *atomic (local) states / conditions* while set E is the set of *atomic (local) transitions/ events*. It is important to note that states and transitions are distinct. *Global (distributed) state* is the case set of conditions holding concurrently while *global (distributed) transition* is the set of events occurring concurrently.

There is also a graphical notation for the elements of Petri nets. State elements are drawn as circles ◯, the transition elements are represented with boxes ▮ and the flow relation is shown as edges ⟶

If $X = B \cup E$ and $x \in X$, as usual ${}^\bullet x$ denotes the set $\{y \in X \,|\, (y,x) \in F\}$; $x^\bullet$ is defined similarly. A net is *pure* when $\forall x \in X \; {}^\bullet x \cap x^\bullet = \varnothing$. A net is *simple* when $(\forall x, y \in X)$ ${}^\bullet x = {}^\bullet y$ and $x^\bullet = y^\bullet$ implies $x = y$. In this study pure and simple nets are used.

A set of conditions $c \subseteq B$ is called a *case*. An event $e \in E$ can occur at a case c when ${}^\bullet e \subseteq c$ and $e^\bullet \cap c = \varnothing$; we say that e is *enabled* at case c. When e occurs at c, the case following c will be $c' = (c \setminus {}^\bullet e) \cup e^\bullet$. When e occurs and transforms case c in c' we say that event e is *fired*. Cases are graphically represented by black *tokens* in the conditions belonging to c.

Example 4.1. In Figure 4.1 firing rule of Petri nets is shown by a simple example.

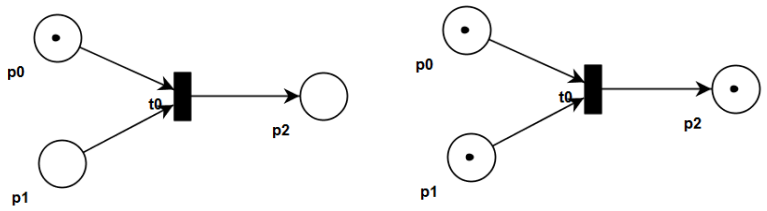

FIGURE 4.2: Demonstration of firing rule: (a) t_0 is not enabled (b) t_0 cannot fire

Example 4.2. Another example shown in Figure 4.2 demonstrates a case that violates firing rule.

In Figure 4.2 (a) t_0 cannot fire since its preconditions are not held. In other words p_1 must hold a token so that t_0 would be enabled. In Figure 4.2 (b) t_0 cannot fire because its postcondition is not empty.

There are many types of Petri Nets including place – transition nets, condition – event nets / elementary nets , colored Petri nets, timed Petri nets and stochastic Petri nets.

In this thesis only the elementary Petri nets are used. In an elementary Petri net system there are conditions, that can either be true or false represented by places and events, that can occur represented by transitions.

In this work elementary Petri nets are used due to the below considerations:

- Elementary nets leave the representation open to the inclusion of low-level signal processing;
- They provide use of basic tools for proving structural properties of the model like invariants analysis at the level of basic signals;
- By using elementary nets one has the properties of interest represented as boolean variables;
- Use of Elementary Petri Nets allows researchers to use powerful model checking tools like LOLA [7] [27].

4.2 Modeling with Elementary Petri Nets

In this work we used elementary Petri nets for modeling DPKG for the formal analysis of the protocol. The reasons for choosing elementary nets are stated in previous section.

Definition 4.2. An elementary net is a 4-tuple

$$EN = (B, E, F, m_0)$$

where (B, E, F) is the underlying net of EN denoted by $und(N)$ and m_0 is the initial marking (initial case) of EN, $inc(EN)$

Let EN be an elementary net system and B_{EN}, E_{EN} and F_{EN} denote respectively the set of conditions in EN, the set of transitions in EN and the set of arcs in EN. Furthermore, $M_{EN} = [m_0 >_{EN}$ the set of markings (cases) of EN.

While $m \in M_{EN}$ and $e_1, e_2 \in E_{EN}$ e_1 and e_2 can be related to each other at m in five ways at least:

1. Sequence:

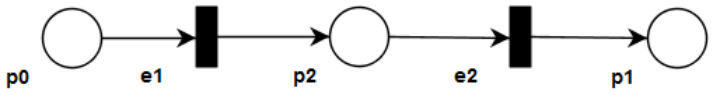

FIGURE 4.3: Sequence of events in a Petri net system

e_1 can occur but e_2 cannot occur until e_1 occurs. e_1 and e_2 are in sequence at m if and only if $m[e_1 >, \neg(m[e_2 >), and m'[e_2 > m'$. Sequence of events is demonstrated in Figure 4.3.

2. Conflict (choice):

e_1 and e_2 can occur individually at m but they cannot occur together at the same time. e_1 and e_2 are in conflict at m if and only if $m[e_1 >, m[e_2 > and \neg(m[e_1, e_2])$. An example of conflict case is displayed in Figure 4.4.

3. Concurrency:

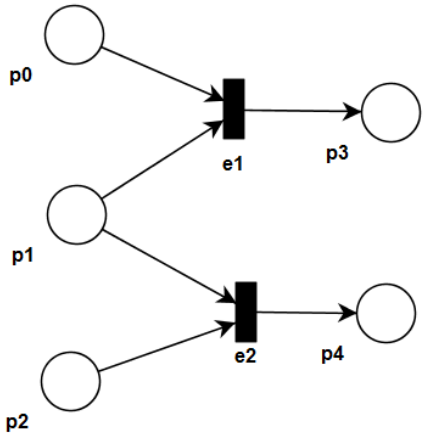

FIGURE 4.4: Conflict case in a Petri net system

e_1 and e_2 can occur at m without interfering each other. No order is specified over the occurrence. e_1 and e_2 can occur at m if and only if $m[e_1, e_2 >$. Concurrency of two events e_1 and e_2 can be seen in Figure 4.5.

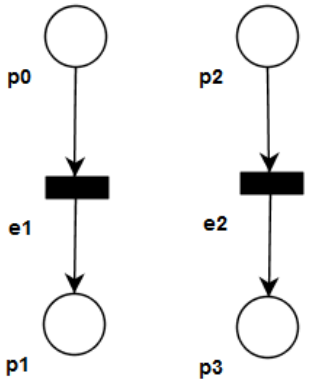

FIGURE 4.5: Concurrency of two events in a Petri net system

In addition to sequence, conflict and concurrency many other characteristics of a distributed system activities can be modeled by elementary Petri nets such as synchronization and mutually exclusion.

4. Synchronization:

In a distributed event system an event may require multiple resources. The resulting synchronization of resources can be captured by transitions of the type shown in Figure 4.6. In the figure, e_1 is enabled only when each of p_1 and p_2 receives a token. The arrival of a token into each of the two places could be the result of a sequence of events elsewhere in the rest of the Petri net model.

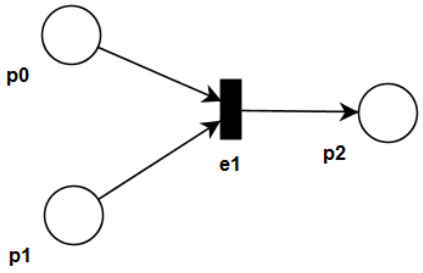

FIGURE 4.6: Case of synchronization in a Petri net system

5. Mutually exclusion:

Two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. Demonstration of this situation in Petri nets is given in Figure 4.7.

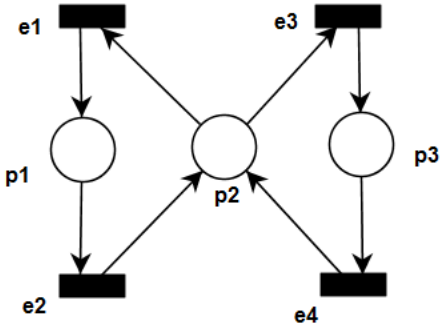

FIGURE 4.7: Two mutually exclusive processes modeled with Petri nets

4.3 Properties of Petri Nets

In this section some of the behavioral and structural properties of Petri nets are considered. Behavioral properties refer to the properties of a Petri net that are dependent on the initial marking of the net and structural properties. Structural properties are only dependent on the structure of the net.

4.3.1 Behavioral Properties

The reachability, boundedness, safety, limitation, liveness, deadlock, reversibility, home state, coverability, persistence and fairness are often referred behavioral properties. For more detailed information referring to [26] is recommended. In the following some of the behavioral properties of Petri nets are given.

When analyzing Petri net systems it is often desirable to know if the given marking m_1 can be reached from the initial marking m_0 or if an undesirable marking m_2 can be avoided. To know the reachability of a marking it is necessary to know if there is a firing sequence that can bring the net to a new marking. This problem is referred as *reachability problem*.

Definition 4.3. A marking m_i is said to be *reachable* from a marking m_0 if there exists a sequence of firings which transforms marking m_0 to m_i.

A marking m_1 is said to be *immediately reachable* from m_0 if firing an enabled transition in m_0 results in m_1.

Possible markings reachable from m_0 for Petri net N is denoted by $R(N, m_0)$.

Another property which is especially useful when the size of the system is wanted to be defined is *boundedness*.

Definition 4.4. A Petri net (N, m_0) is *k-bounded* or simply *bounded* if $m(p) \leq k$ for every place p and every marking $m \in R(m_0)$.

Definition 4.5. A Petri net is *safe* if it is k-bounded and $k = 1$.

By verifying that the net is bounded or safe it is guaranteed that there will be no overflows in the buffers or registers of the system, No matter what firing sequence is taken.

The *liveness* property is closely related to the complete absence of deadlocks in a system. A Petri net (N, m_0) is said to be live if, no matter what marking has

been reached from m_0 , it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen.

Definition 4.6. A Petri net is said to be *live* if for all transitions there is a way to fire transition in any marking m' reachable from the initial marking m_0 i.e. $\forall m' \in R(m_0)$ and $\forall e \in E, \exists m \in R(m')$ such that e is enabled in m .

Definition 4.7. A marking m' reachable from the initial marking m_0 is a *deadlock* if none of the transitions of the Petri net is enable in m' .

Reversibility is one of the often used properties demanded by many applications. In reversible net one can always get back to the initial marking or state.

Definition 4.8. A Petri net (N, m_0) is said to be *reversible* if for each marking $m \in R(m_0), m_0$ is reachable from m .

In many applications, it is not necessary to get back to the initial state as long as one can get to some (home) state.

Definition 4.9. A marking m' is said to be a *home state* if, for each marking $m \in R(m_0), m'$ is reachable from m.

These three properties boundedness, liveness and reversibility are independent from each other. A reversible net can be live or not live and bounded or not bounded.

Another property is coverability and it is closely related to 1-liveness (potential firability).

Definition 4.10. A marking m in a Petri net (N, m_0) is said to be *coverable* if, there exists a marking $m' \in R(m_0)$ such that $m'(p) \geq m(p) forall p \in P$.

4.3.2 Structural Properties

Structural properties are the properties that depend on the topological structures of Petri nets. They are independent of the initial marking m_0. Some of the structural properties of Petri nets are listed below:

- *Structural Liveness:* A Petri net N is said to be structurally live if there exists a live initial marking m_0 for N.
- *Controllability:* A Petri net N is said to be completely controllable if any marking is reachable from any other marking.

- *Structural Boundedness:* A Petri net N is said to be structurally bounded if it is bounded for any finite initial marking m_0.
- *Repetitiveness:* A Petri net N is said to be repetitive if there exists a marking m_0 and a firing sequence from m_0 such that every transition occurs infinitely often.
- *Consistency:* A Petri net N is said to be consistent if there exists a marking m_0 and a firing sequence from m_0 back to m_0 such that every transition occurs at least once.

4.4 Analysis of Elementary Petri Nets

Methods for analyzing Petri nets can be classified in three groups as the *coverability / reachability tree* method, *matrix equation* approach and *reduction or decomposition* techniques. Coverability / reachability tree method involves the enumeration of all reachable markings or their coverable markings instead. This method is limited to small nets due to the complexity of the state space explosion. Boundedness, safeness, liveness and reachability are some of the properties can be studied by using coverability trees. However, the reachability and liveness problems cannot be solved by the coverability tree method alone.

For a bounded Petri net, the coverability tree is equal the reachability tree since it contains all possible markings.

Definition 4.11. The *coverability graph* of Petri net (N, m_0) is labeled directed graph $G = (V, E)$. Its node set V is the set of all distinct labeled nodes in the coverability tree and the arc set E is the set of arcs labeled with single transition t_k representing all possible single transition firings such that $m_i[t_k > m_j$, where m_i and m_j are in V. For a bounded Petri net, the coverability graph is referred to as the *reachability graph*, because the vertex set V becomes the same as the reachability set $R(m_0)$.

Definition 4.12. A *reachability graph* of a Petri net N is a directed graph $G = (V, E)$, where $v \in V$ represents a class of reachable markings; $e \in E$ represents a directed arc from a class of markings to the other class of markings.

A reachability graph is also called *occurrence graph* or *state space*. The reachability graph demonstrates a better performance than the reachability tree.

Although a Petri net is finite, the set of its reachable markings is not always finite. For instance, when a Petri net is not safe or bounded, its number of tokens can be infinite, thus the set of reachable markings being infinite. This situation is not likely to happen in elementary Petri nets due to its structure letting a place/condition to have only one token or no token at all. However, the reachability graph analysis still can be applied on the small nets due to the state space explosion. In this study the nets are divided into smaller ones in the aim of creating feasible reachability graphs for analysis. The reachability graphs are used to determine and analysis the safeness, boundedness, conservation, reachability/coverability and liveness properties of Petri nets. The major reason for using reachability graph analysis in this study is to determine whether a specific marking/case is reachable where the specific marking demonstrates a situation which violates the security of a system or a situation that has to be reached in order to perform a protocol successfully.

Matrix equations and reduction techniques are powerful but they are only applicable for special subclasses of Petri nets or for some special situations.

For more detailed information about analysis methods it is recommended to read [26].

4.5 Modeling and Analyzing Cryptographic Protocols with Elementary Petri Nets

Using cryptographic techniques to provide network or information security can end up with potential loopholes for an intruder to compromise the system security no matter how strong is the cryptographic algorithm implemented is if the protocol is improperly designed. Hence, the security of these applications does not only depend on the strength of the underlying cryptographic algorithm but also on the security and correctness of the protocols.

The purpose of cryptographic protocol modeling is to provide a formal representation of protocol systems as well as facilitate formal analysis on these systems. Elementary Petri nets provide a formal basis on which we can investigate security properties of the protocols and search for security flaws and it also provides structural property analysis such that liveness, safeness and boundedness.

As it is also stated in the Chapter 1, one of the reasons for choosing elementary Petri nets for this study is because it allows us for the use of powerful model checking tools like LoLA [7].

Chapter 5

Formal Model of Distributed Private Key Generation for Identity Based Cryptography

5.1 Modeling Distributed Private Key Generation for Identity Based Cryptography

In this section Petri net model of Distributed Private Key Generation (DPKG) for Identity Based Cryptography is presented in the aim of providing a formal model of the system and analyzing the security properties of DPKG protocol. In Section 5.1.1, a simple model for the whole DPKG is represented for a better understanding of the process. In the following sections, more detailed models are presented and being analyzed by means of structural properties as well as behavioral properties of the nets. An intruder model is also provided for analyzing the behavior of the system under an attack against the protocol. PIPE2 [28] and LoLA [7] [27] tools are used for creating and analyzing the nets.

5.1.1 Modeling Overall Distributed Private Key Generation and Analysis of the Model

To begin with, an overall model for DPKG is presented in Figure 5.1. The Petri net represents a $(3,2)$ DPKG system with three Private Key Generator (PKG) nodes, a

bulletin board and a client. In the beginning, three PKG nodes wait for a distribution and a bulletin board stores and broadcasts the system parameters. Together they generate a master key pair consists of master private key s and master public key (Pub) in a distributed manner. In the model, it can be seen that a client applies to two PKG nodes to extract a private key regarding the fact that it is a $(3,2)$ system. Having two signed pieces as $s_i H(ID_A)$ (i = 1, 2, 3 here) from two distinct PKG nodes, client A can extract its private key by evaluating the formula 2.17. It is important to note that there are 3 pieces of master private key and none of the PKG nodes has the whole key s. Each of them has a share for it $share_i$ and a share for the master public key Pub_i. In the model, `share1`, `share 2` and `share 3` represents PKG1, PKG2 and PKG3 holding the shares. The reason for duplicating those places with `p11`, `p12` and `p13` is to represent that the shares are not consumed after a client applies to the PKG node without using a self loop which would cause the net to loose its simplicity. The simplicity of a Petri net is described in Section 4.1. For more information about elementary Petri nets it is recommended to refer to [25].

The net is constructed in a way to represent the basic concept of the DPKG. It is also taken into consideration that both the distribution of the shares and the extraction of the client's private key has to be repeatable. One can easily notice that the net can be 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.

5.1.2 Model of Setup and Distribution Step

The net shown in Figure 5.2 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 net starts with the case that places with labels `idle`, `initCanStart` and `BBidle` have one token each, while the other places have no tokens. Since the model is constructed by using elementary net systems, it is possible to claim that `idle`, `initCanStart` and `BBidle` local conditions are logically true while all the other local conditions are logically false. The only event that can be fired is `initializePKGnodes`.

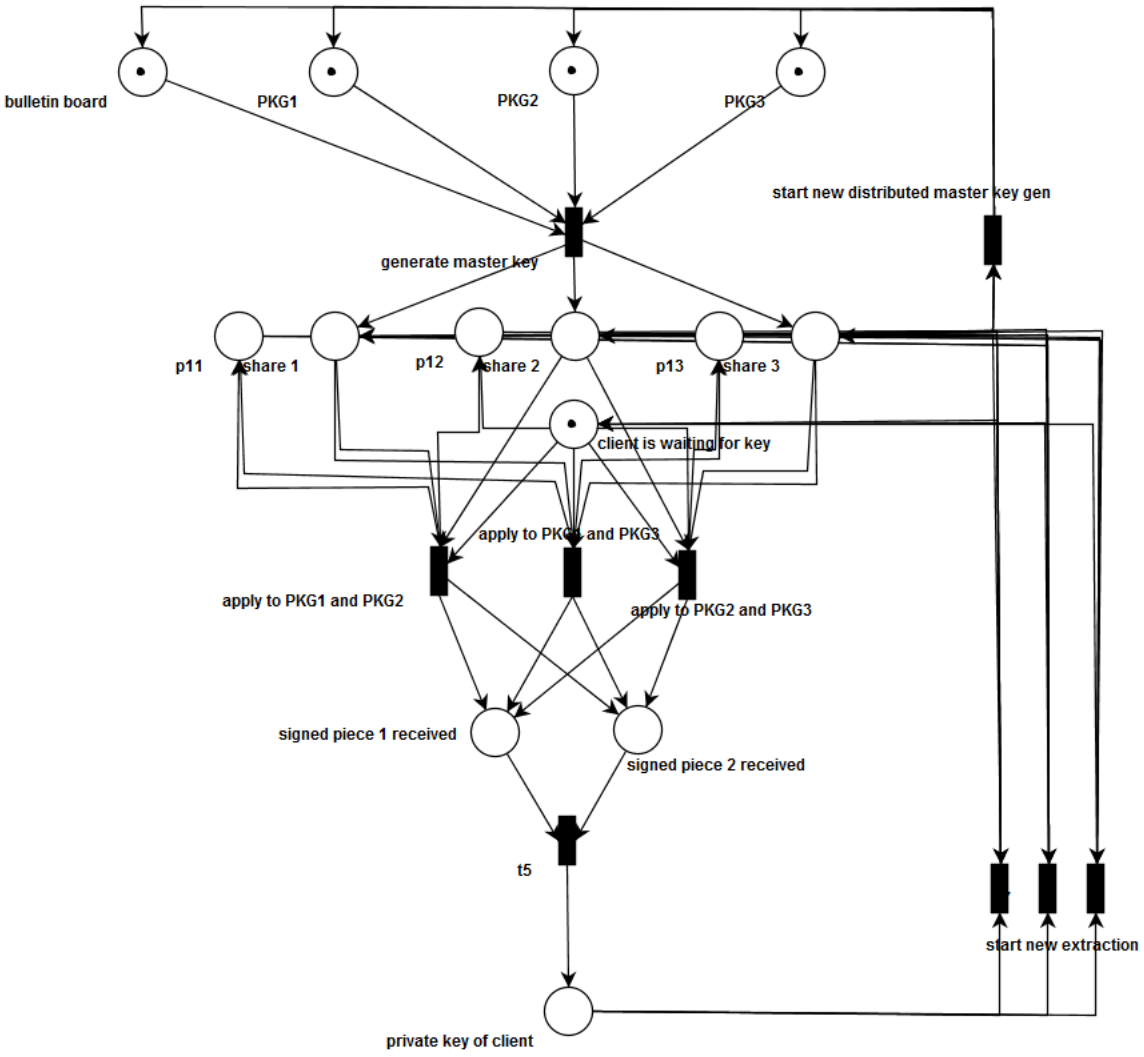

FIGURE 5.1: A net representing simple overall look to the DPKG

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 bulletin board. Only one of the flows will be explained here since the other two are alike. The event `choseP1andSendToBB` 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 broad casted. a_{il} is the l^{th} coefficient of i^{th} polynomial. It is important to note that PKG1 does not send the polynomial it only sends the commitments. When this event is fired, the local states `P1readyForSubshares` and `commitmentsP1` become true.

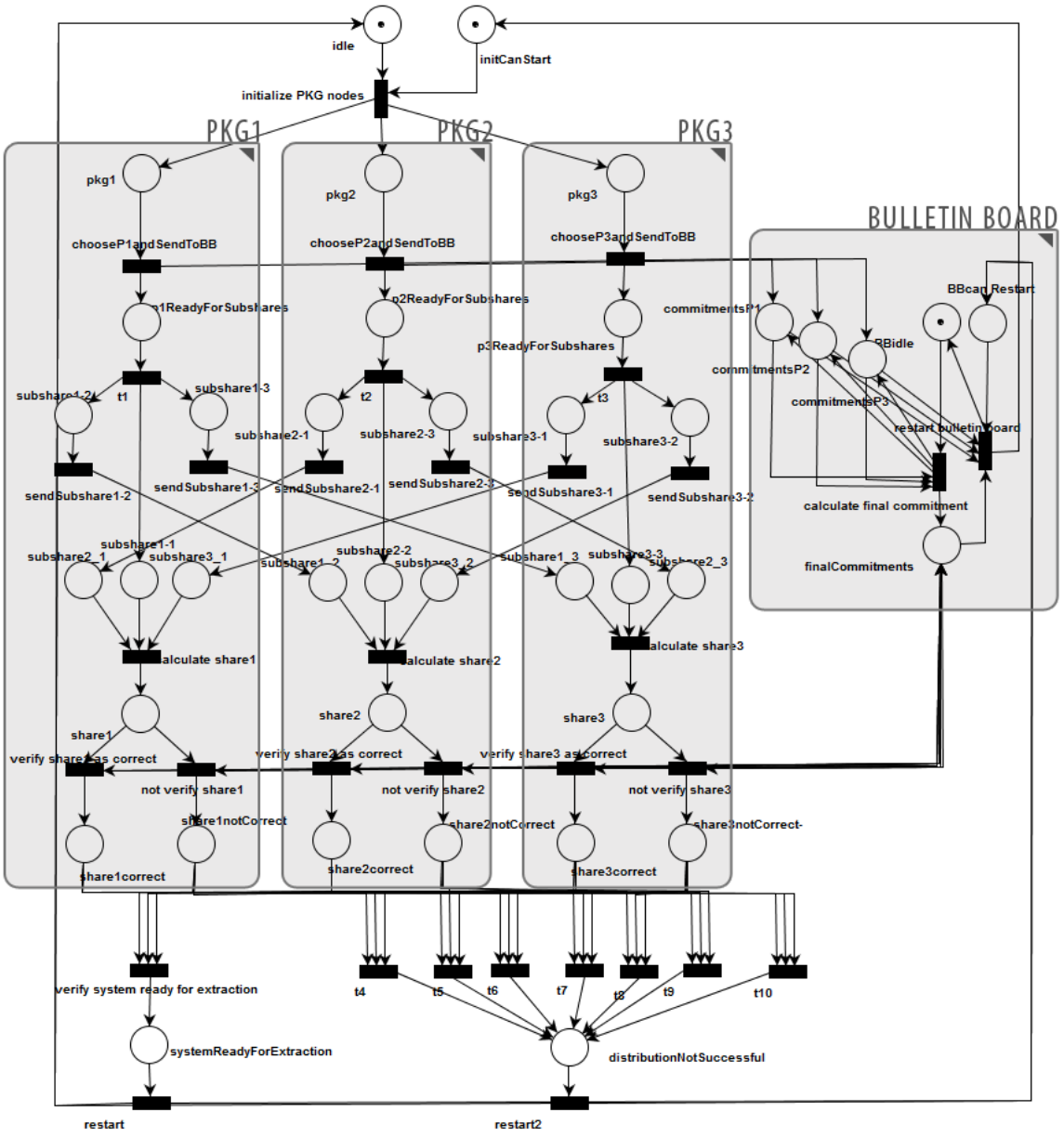

FIGURE 5.2: Petri net model of distribution step of distributed private key generation.

Independently, when all three PKG nodes send the commitments, all four precondition states of `calculate final commitment` event become true so the event is enabled. After `p1ReadyForSubshares` becomes true, the event `calculate subshares` is enabled. The firing of this event 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 their selves 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 events are already fired and `subshare 2_1`, `subshare1-1` and `subshare3_1` local states are all true, The event `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 `share 1` becomes true, if final commitment is calculated, two events are enabled: `verify share1 as correct` and `not verify share1`. 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 events `t4, t5, t6, t7, t8, t9, t10` becomes enabled. Firing of this event consumes the remaining tokens to make system ready to restart. The arrows in this part are not shown separately to prevent the complicity.

Only one of the local states `systemReadyForExtraction` and `distributionNotSuccessful` can be true and the system can be restarted both after a success and fail. In any of the restart cases 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.

5.1.3 Model of Key Extraction Step

The Petri Net model of the extraction part is shown in Figure 5.3. Extraction starts after all three shares are ready and verified as correct and there is a client requesting a private key. The places `PKG1 ready`, `PKG2 ready` and `PKG3 ready` in extraction model overlaps with `share1correct`, `share2correct` and `share3correct` in the distribution model.

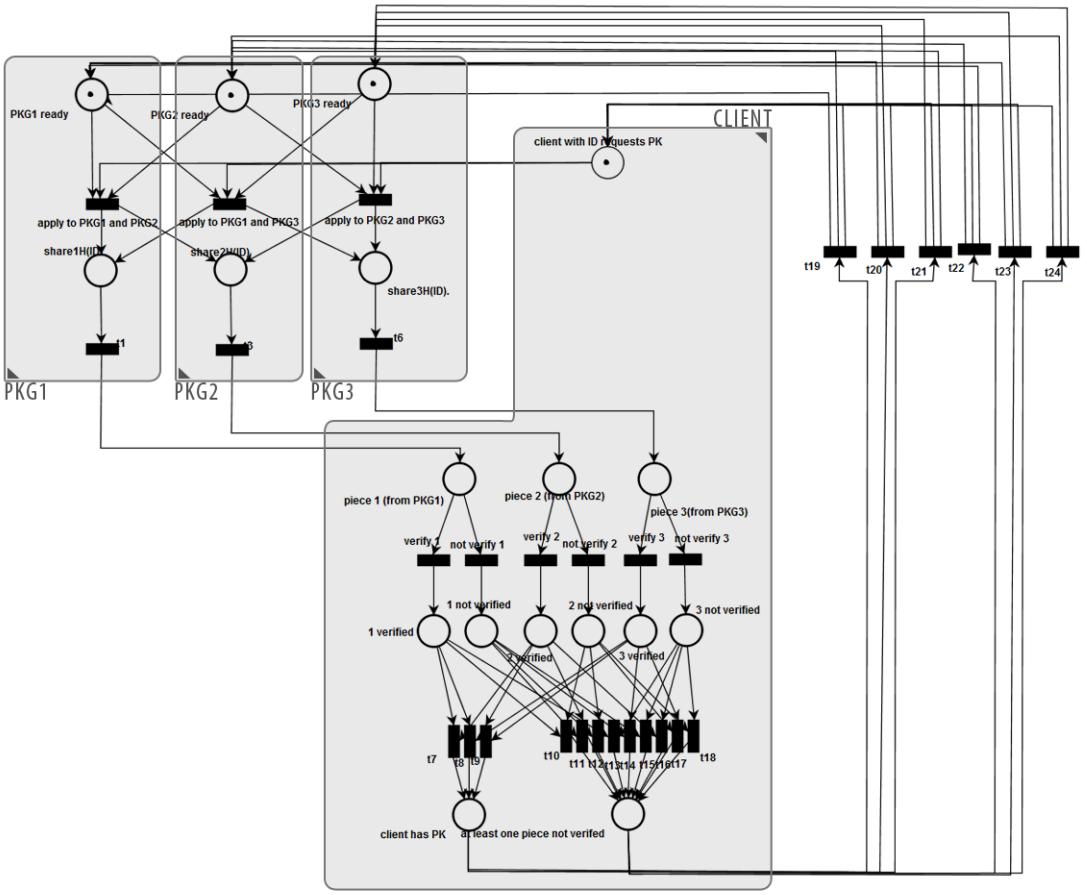

FIGURE 5.3: Petri net model of extraction step of distributed private key generation.

We could have combined these two models to make the model complete, but in this case the model would be too big to analyze. 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 events (`apply to PKG1 and PKG2`, `apply to PKG1 and PKG3` and `apply to PKG2 and PKG3`) are enabled but only one can be fired.

For instance if `apply to PKG1 and PKG2` event is fired, `share1H(ID)` and `share2H(ID)` become true. The firing of the event actually means that PKG1 does elliptic curve scalar multiplication with share1 and hashed ID of client and PKG2 does the same. In the net sending the result to the client is done by events `PKG1 sends to client` and `PKG2 sends to client`.

After sending, the flow goes on in the client part. In this case, `piece1` and `piece 2` states become true. For each state representing a piece there are two events 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 `at least one piece not verified` becomes true. After both successful and unsuccessful states the system can be restarted for a new extraction. There are six restart events since each one is enabled in a different ending state and the old tokens are needed to be consumed to go back to the starting state.

5.1.4 Analysis on the DPKG model

In this section, we first introduce a structural analysis of the DPKG model for Identity based Cryptography then continue with a more detailed analysis. 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. Elementary Petri net model enables us to do a security property analysis for the protocol.

The first basic analysis on the models in Figure 5.2 and Figure 5.3 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 1: 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 BulletinBoard 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 5.2, we see the result for the formula is false which means the global state is not reachable in the model.

FORMULA 2: 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 5.2 is checked for the formula.

FORMULA 3: EXPATH EVENTUALLY (clientHasCorrectKey = 1 AND atLeastOnePieceNotVerified = 1)

The last formula checks the existence of a global state in the extraction model shown in Figure 5.3. 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.

5.1.5 Intruder Model

As in the traditional attack models, an attacker is represented by an insecure channel. The channel can consequently substitute packets, re-use old keys and play all the possible man-in-the-middle tactics. Two main goals are foreseen: 1) to block the process by exploiting the weaknesses of the DPKG protocol; 2) to extract information about the system.

We introduce an intruder model which sneaks into the channel during the distributed key generation. Since the model is too complex to analyze, it is split into two parts as distribution and extraction.

The first intruder model is shown in Figure 5.4. 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 in the model listens to the channel between PKG1 and PKG2 having also the

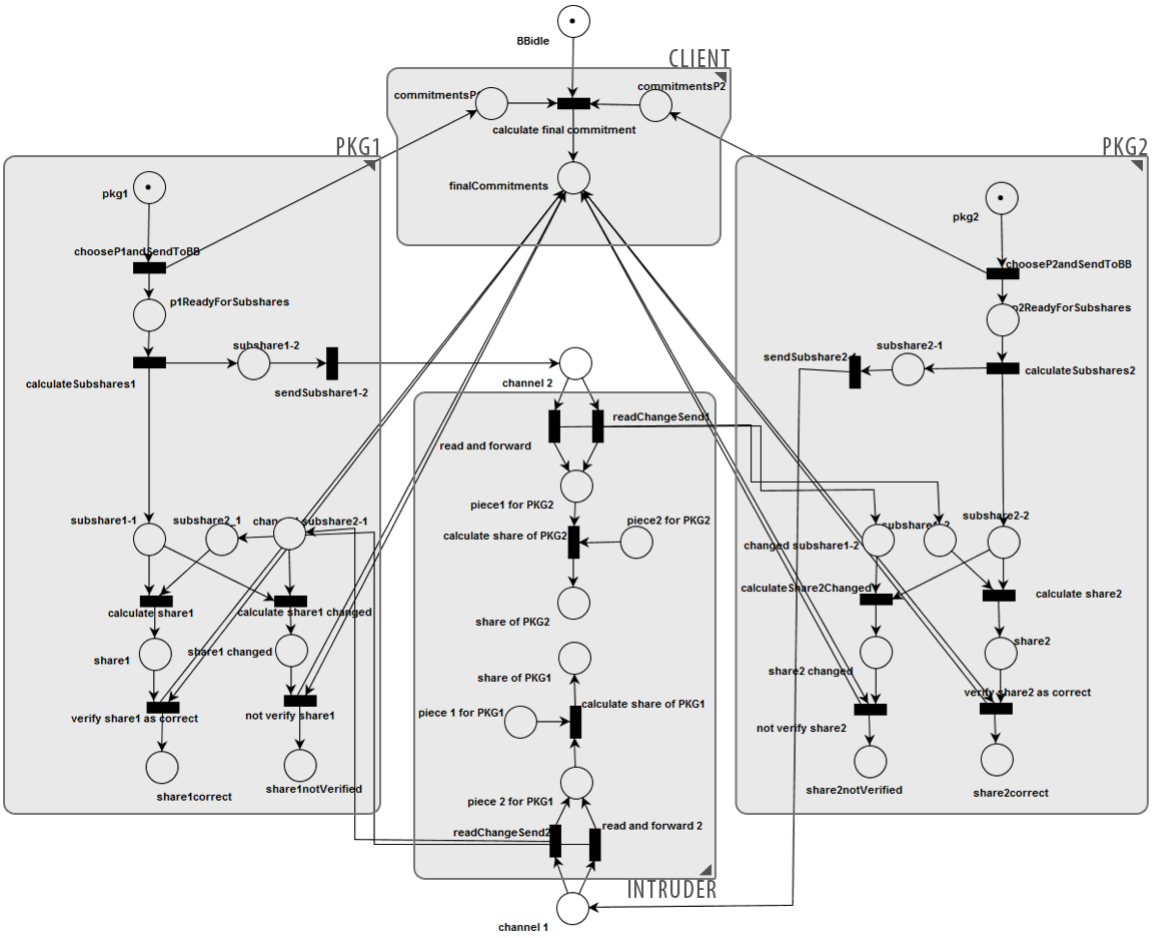

FIGURE 5.4: Petri net model of an intruder in the distribution step of DPKG.

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 on 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 produces two subshares. To explain with an example, in order to have a share for the master private key PKG1 needs to obtain two subshares. One is the subshare created by PKG1 and the other is the subshare created by PKG2 for PKG1 as $< i, P(i) >$ where P is the secret polynomial of the PKG2.

The second intruder model representing the extraction step with an intruder is shown in Figure 5.5. 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.

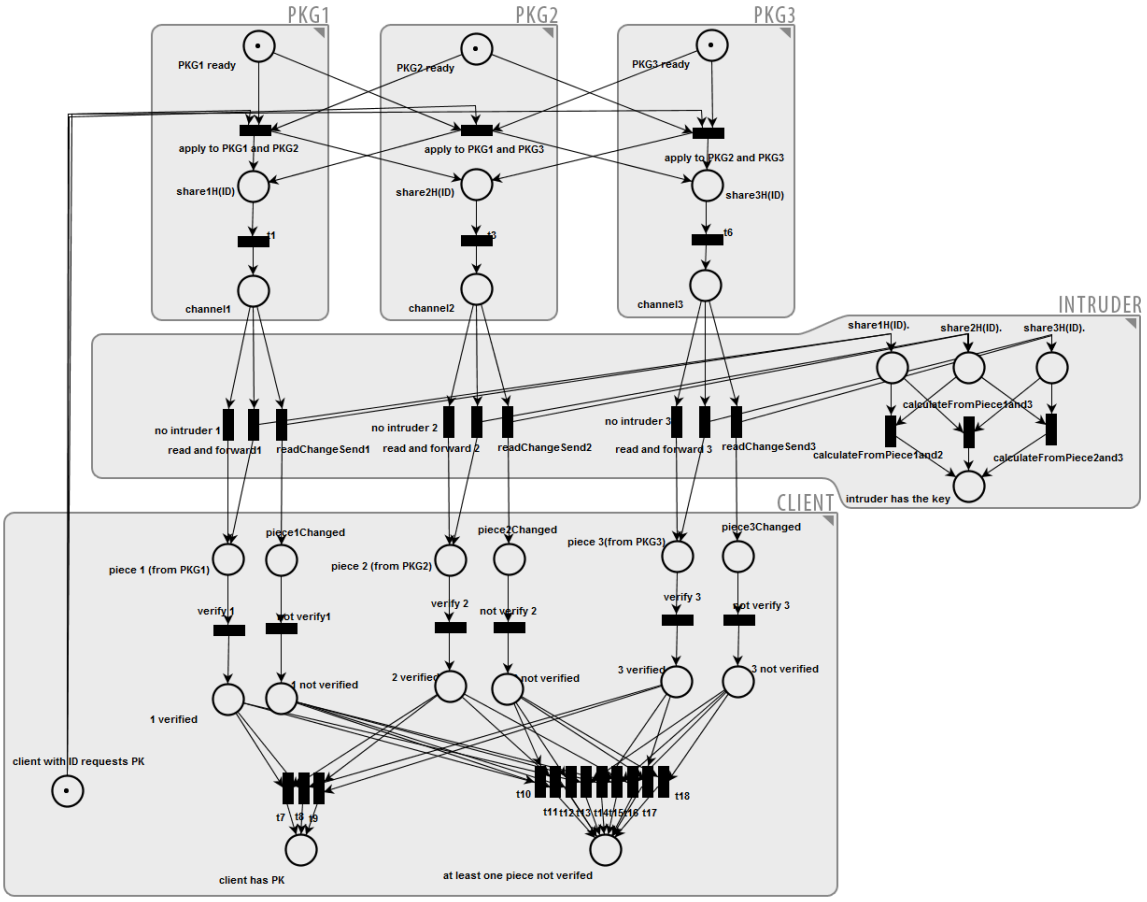

FIGURE 5.5: Petri net model of an intruder in the extraction step of DPKG.

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.

5.1.6 Analysis on the Intruder 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 5.1.4.

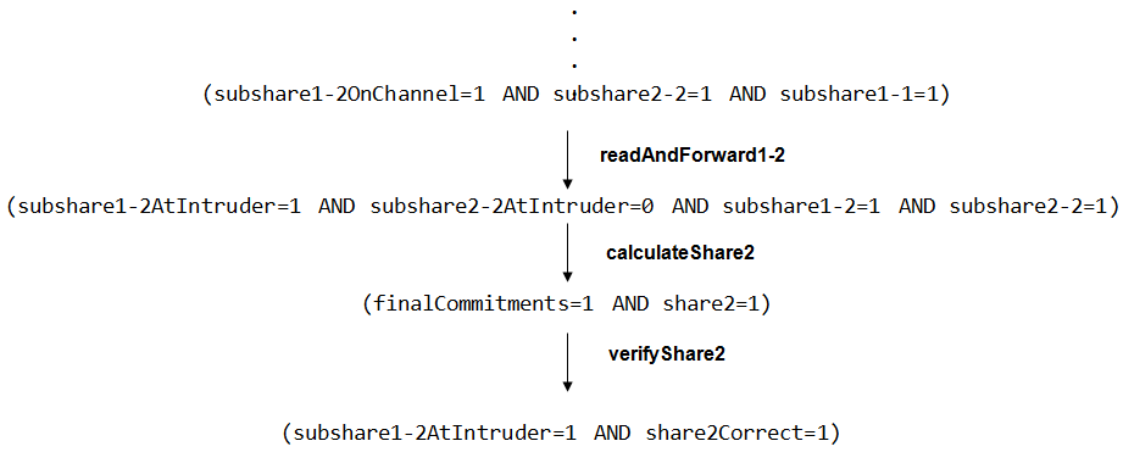

FIGURE 5.6: The sequence of events for making Formula 5 true

FORMULA 4: EXPATH EVENTUALLY ((subshare1-2AtIntruder = 1 AND subshare2-2AtIntruder = 1) OR (subshare2-1AtIntruder = 1 AND subshare1-1AtIntruder = 1))

With this formula we ask 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 5.4 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 5: EXPATH EVENTUALLY (subshare1-2AtIntruder = 1 AND share2Correct = 1)

The next formula checks if there exists a global state in which intruder reads the subshare 1-2 but PKG2 does not know it. The result is true since this situation is likely to happen when the intruder only reads and forwards the subshare without changing it. It is not fatal until all the required subshares are read by the intruder. In Figure 5.6 the path of firings leading the mentioned global state is given.

FORMULA 6: 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 pieces she received is not verified. When the model shown in Figure 5.5 is checked for this formula we see that the result is false. This scenario could be correct if the result of verification done by using bilinear

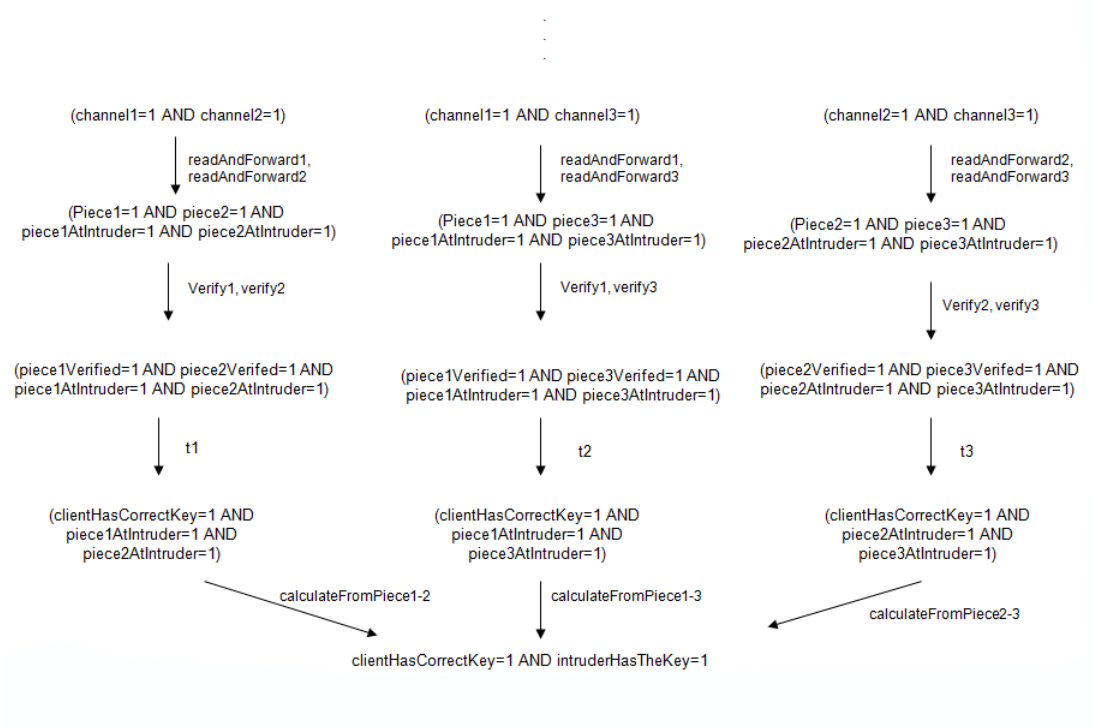

FIGURE 5.7: The sequence of events for making formula 7 true

cryptographic pairings in Algorithm 3 is correct while the parameter of the pairing function is wrong. The details about cryptographic pairing functions can be seen in [29].

FORMULA 7: 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 about it.

Figure 5.7 shows the path of firings which make formula 7 true.

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 shown in Figure 5.5 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.

Chapter 6

Results and Conclusion

In this thesis it is aimed to create a formal model for Distributed Private Key Generation protocol and analyze the behavior of it for proving its properties.

To this end, a basic model of the protocol is built by using Elementary Net Systems. This model allows for the first reflections about the protocol by means of traditional structural analysis techniques. Then the work is extended by including an intruder in the model aiming at analyzing the behavior of the protocol under an attack 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 Section 5.1.5 are not to be achieved. This thesis focuses on the protocol of distributed private key generation, thus we cannot say anything about the flaws that can occur due to the wrong use of the keys for encryption, decryption and signature.

In addition to the formal analysis, a practical analysis of the protocol is also performed by implementing DPKG on industrial control systems including use of the generated keys for providing the system with authenticity by identity based signatures. After several tests and analysis it is concluded that identity based cryptographic approach with distributed private key generation is advantageous for systems like SCADA since it makes the use of certificates and certification authorities unnecessary as well as it does not require use of centralized repositories for storing public keys. Excluding the certificates from the system makes the encryption

& decryption and signature & verification processes faster and easier in terms of managing the public keys.

Bibliography

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 47–53.

[2] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., vol. 2139. Springer, 2001, pp. 213–229.

[3] C. A. Petri, "Fundamentals of a theory of asynchronous information flow," in *IFIP Congress*, 1962, pp. 386–390.

[4] B. B. Nieh and S. E. Tavares, "Modelling and analyzing cryprographic protocols using petri nets," in *AUSCRYPT*, ser. Lecture Notes in Computer Science, J. Seberry and Y. Zheng, Eds., vol. 718. Springer, 1992, pp. 275–295.

[5] Y. Xu and X. Xie, "Modeling and analysis of security protocols using colored petri nets," *JCP*, vol. 6, no. 1, pp. 19–27, 2011.

[6] R. Bouroulet, R. R. Devillers, H. Klaudel, E. Pelz, and F. Pommereau, "Modeling and analysis of security protocols using role based specifications and petri nets," in *Petri Nets*, ser. Lecture Notes in Computer Science, K. M. van Hee and R. Valk, Eds., vol. 5062. Springer, 2008, pp. 72–91.

[7] K. Schmidt, "Distributed verification with lola," *Fundam. Inform.*, vol. 54, no. 2-3, pp. 253–262, 2003.

[8] J. B. Fraleigh, *A first course in abstract algebra*, 5th ed., L. Rosatone, Ed. Addison-Wesley Pub. Co, 1994.

[9] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan. 1987.

[10] V. S. Miller, "Use of elliptic curves in cryptography," in *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*. New

York, NY, USA: Springer-Verlag New York, Inc., 1986, pp. 417–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=18262.25413

[11] J. Hoffstein, J. Pipher, and J. Silverman, *An Introduction to Mathematical Cryptography*, ser. Undergraduate Texts in Mathematics. New York, NJ, USA: Springer, 2008. [Online]. Available: http://books.google.com/books?id=62PYj63QalkC

[12] V. S. Miller, "The weil pairing, and its efficient calculation," *J. Cryptol.*, vol. 17, no. 4, pp. 235–261, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1007/s00145-004-0315-8

[13] S. Edixhoven, "Le couplage weil: de la geometrie a l arithmetique," 2002.

[14] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] S. M. Benny Chor, Shaft Goldwasser and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *proceedings of 6th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 383–395.

[16] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, ser. SFCS '87. Washington, DC, USA: IEEE Computer Society, 1987, pp. 427–438. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1987.4

[17] A. Kate and I. Goldberg, "Asynchronous distributed private-key generators for identity-based cryptography," *IACR Cryptology ePrint Archive*, vol. 2009, p. 355, 2009.

[18] R. L. Igor Nai Fovino, Marcelo Masera, "Ict security assessment of a power plant, a case study," in *Second Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, Arlington, USA, 2008.

[19] M. IDA. Modbus application protocol specification. [Online]. Available: http://www.modbus.org/specs.php

[20] M. P. P. Huitsing, R. Chandia and S. Shenoi, "Attack taxonomies for the modbus serial and tcp protocols," in *Second Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, 2008.

[21] R. Leszczyna, I. N. Fovino, and M. Masera, "Maisim: mobile agent malware simulator," in *SimuTools*, S. Molnár, J. R. Heath, O. Dalle, and G. A. Wainer, Eds. ICST, 2008, p. 35.

[22] [Online]. Available: http://crypto.stanford.edu/pbc/

[23] T. modbus consortium. (2012, Jan) The modbus organization web page. [Online]. Available: http://www.modbus.org/

[24] C. A. Petri, "Kommunikation mit Automaten," Ph.D. dissertation, Institut für instrumentelle Mathematik, Bonn, 1962.

[25] M. Nielsen and D. Simpson, Eds., *Application and Theory of Petri Nets 2000, 21st International Conference, ICATPN 2000, Aarhus, Denmark, June 26-30, 2000, Proceeding*, ser. Lecture Notes in Computer Science, vol. 1825. Springer, 2000.

[26] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989. [Online]. Available: http://dx.doi.org/10.1109/5.24143

[27] [Online]. Available: http://www.informatik.uni-rostock.de/tpp/lola/

[28] [Online]. Available: http://pipe2.sourceforge.net/

[29] G. S. Ian F. Blake and N. P. Smart, *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*, I. F. Blake, Ed. New York, NY, USA: Cambridge University Press, 2005. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=JQv2ySOsQkQC&oi=fnd&pg=PR9&dq=Advances+in+Elliptic+Curves+Cryptography&ots=AIHCUP_BxK&sig=v1DQu-8PBJmmLywGSnMRq-LyZYE