Quantitative Analysis of a Certified E-mail Protocol in Mobile Environments: A Probabilistic Model Checking Approach

S. Basagiannis[a,*], S. Petridou[a], N. Alexiou[a], G. Papadimitriou[a], P. Katsaros[a]

[a] *Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*

Abstract

Formal analysis techniques, such as probabilistic model checking, offer an effective mechanism for model-based performance and verification studies of communication systems' behavior that can be abstractly described by a set of rules i.e., a protocol. This article presents an integrated approach for the quantitative analysis of the Certified E-mail Message Delivery (CEMD) protocol that provides security properties to electronic mail services. The proposed scheme employs a probabilistic model checking analysis and provides for the first time insights on the impact of CEMD's error tolerance on computational and transmission cost. It exploits an efficient combination of quantitative analysis and specific computational and communication parameters, i.e., the widely used Texas Instruments TMS320C55x Family operating in an High Speed Downlink Packet Access (HSDPA) mobile environment, where multiple CEMD participants execute parallel sessions with high bit error rates (BERs). Furthermore, it offers a tool-assistant approach for the protocol designers and analysts towards the verification of their products under varying parameters. Finally, this analysis can be also utilized towards reliably addressing cost-related issues of certain communication protocols and deciding on their cost-dependent viability, taking into account limitations that are introduced by hardware specifications of mobile devices and noisy mobile environments.

Keywords: Certified e-mail, probabilistic model checking, CTMC, mobile environments

1. Introduction

During the last decades, the fact that a number of communication protocols have been published with security flaws [1, 2, 3] urges the protocol designers

*Corresponding author

Email addresses: basags@csd.auth.gr (S. Basagiannis), spetrido@csd.auth.gr (S. Petridou), nalexiou@csd.auth.gr (N. Alexiou), gp@csd.auth.gr (G. Papadimitriou), katsaros@csd.auth.gr (P. Katsaros)

to "strengthen" their products with additional cryptographic mechanisms (e.g., public key cryptography) that secure and guarantee the safe completion of protocol's sessions. But, the tradeoff of gaining in security is losing in terms of computational cost. Increased computational cost, in turn, entails that a protocol cannot be adopted by low-cost hardware equipment, such as mobile devices. Moreover, when discussing protocols' cost in mobile environments, parameters that determine transmission cost, such as the high bit error rate (BER), should be taken into consideration. However, predicting computational and transmission costs is often impossible, due to the presence of multiple participants executing parallel sessions. For example, nowadays, the widespread use of wireless and mobile communications [4, 5, 6, 7] along with the services and applications supported in new generation's mobile devices [8] entails low-cost infrastructure operating in noisy environments. In fact, current WWANs (Wireless Wide Area Networks) are primarily based on second (2G) and third (3G) generation mobile technologies such as GSM (Global System for Mobile Communications) and HSDPA (High Speed Downlink Packet Access), respectively [9, 10, 11]. Thus, it is fundamental to define the conditions under which a secure protocol is suitable for mobile environments.

Nowadays, research in formal methods leads to the development of verification techniques that facilitate the early detection of software or hardware defects in Information and Communication Technology (ICT) systems [12]. Moreover, formal analysis techniques, such as probabilistic model checking, are considered to be an effective way for studying security failures in communication systems, since they can discover design flaws in protocols.

In this paper we propose the use of probabilistic model checking [13] to analyze the Certified E-mail Message Delivery (CEMD) protocol [14, 15]. Current work aims at providing a quantitative analysis in mobile environments. Nowadays, one of the most dominant applications used by mobile devices, e.g., smart phones and PDAs, is the e-mail delivery service [8]. Given the popularity of e-mail service and the current trend for secure communications, analyzing the cost of security in e-mail service is an important issue. CEMD protocol provides all the anticipated security properties, i.e., fairness, confidentiality, timeliness and TTP invisibility [14] that an e-mail protocol ought to support. Moreover, it is a well accepted protocol in related bibliography based on the well known ANR protocol [15].

An abstract representation of the proposed analysis is presented in Fig. 1. As shown in the inner circle our analysis preserves the security properties, i.e., fairness, timeliness, confidentiality and TTP invisibility, of CEMD. The CEMD protocol is modeled as a Continuous-Time Markov Chain (CTMC) [16], while its properties are expressed as Continuous Stochastic Logic (CSL) formulas [17]. The PRISM framework [18] performs automated analysis of the CEMD protocol and verifies the security guarantees it provides. Then, the aforementioned CTMC model is used for the quantitative analysis of the protocol's cost-related properties. These properties are twofold: computational cost imposed by low-cost participants, i.e., mobile devices, and transmission cost due to high BER in mobile environments. As a result, the proposed study provides answers for

whether or not the CEMD protocol can assure security guaranties for low-cost participants involved in multiple protocol sessions and operating in error-prone environments, and if so, at what cost. Thus, the next two circles of Fig. 1 represent the last contribution of our analysis which takes into consideration that CEMD protocol operates in noisy environments such those of mobile communications and involves multiple participants executing parallel protocol sessions. The outer circle of Fig. 1 represents the proposed quantitative analysis incorporating all the above components.

1.1. Contribution

To the best of our knowledge, this is the first work that applies probabilistic model checking for the quantitative analysis of a protocol under specific computational and communication parameters. The current paper considers the widely used Texas Instruments TMS320C55x Family operating at $200MHz$ [19, 20]. Although, nowadays, mobile processors' speed up to $1GHz$, there are many CPU manufacturers who support products working on different modes. For example, Intel launched a Mobile Pentium III processor with two working modes, namely full power mode at $800MHz$ and low power mode at $650MHz$ [21]. This characteristic is very important, since fundamental communication components, e.g., protocols, and their mechanisms, e.g., cryptographic operations, can be developed in a way that exploits their low power mode feature. This entails that protocol designers and analysts can verify their products considering them as operating in low power mode. In this way protocols and security services will be always fully supported contrary to other mobile functionalities in which energy scale-down techniques can be applied e.g., display scale down optimizations (background half or fully dim). Consequently, this paper exploits CPUs at the low corner of $200MHz$, since it is significant for CPU to provide critical services when it operates in low power mode. Works in [8, 21, 22, 23] also pinpoint the necessity of minimizing CPU energy consumption in mobile devices.

Thus, in this work, authors consider computational cost in line with the CPU cycles and the time required by the aforementioned mobile processors to perform RSA operations, i.e. encryption and decryption processes, required by the CEMD protocol. This actually means that the proposed computational cost analysis considers battery life through CPU cycles. Although display and applications, e.g., camera, mp3 and games, consume a great portion of battery life, it is found that CPU and memory are the dominant consuming subsystems in 3G mobile devices [8]. In fact, the CPU power is consumed due to instructions' execution and their fetching from the memories or caches. This in conjunction with the fact that security protocols embed cryptographic mechanisms that, nowadays, require increasing bit size keys, e.g., greater than 512 bits, results in great deal of CPU expenditure.

The communication parameters that the proposed analysis takes into account is the presence of BER in mobile networks, which constitutes them as noisy environments, and the multiple CEMD participants executing parallel sessions. The values of BER in mobile communications are considerable high

compared to those in wired networks and, thus, they should be taken into account. Even in cases where protocols such as HSDPA are adopted the values of BER vary from 10^{-6} to 10^{-3} [10].

Thus, the contribution of the current analysis is that it combines computational and communication parameters, met at current mobile devices and networks, in order to put them into a quantitative analysis which studies an e-mail protocol offering security services. Results derived from this analysis can be exploited by hardware designers who aim to support provision of CEMD protocol services. Given the environment where their products will operate as well as their hardware specifications, they can launch a similar analysis with specific parameters in order to obtain an estimate of the anticipated computational and transmission cost. This information in conjunction with the time that protocol's sessions require to be completed will provide answers about the feasibility of executing the CEMD protocol in hardware with specific characteristics. But, besides this restrictive perspective, the described analysis can be adopted in whatever combination of software and hardware. Thus, the perspective of this work is to provide a methodology for protocol designers and analysts towards the verification of their products under varying parameters.

The remainder of this paper is organized as follows. Section 2 provides a review of related studies, in order to point out the novelty of the proposed analysis approach. Section 3 is a brief introduction to the probabilistic model checking principles based on CTMC models and CSL logic. The CEMD protocol and the PRISM CTMC model along with the assumptions introduced for the proposed quantitative analysis are presented in Section 4. Section 5 discusses the results derived from the analysis as well as the overall usability and potential impact of them. Conclusions and future work insights are given in Section 6.

2. Related Work

Given the widespread use of wireless and mobile communications [4, 5, 6, 7], it is essential for protocol designers to verify the security properties [24] that they are supposed to provide as well as to quantify their cost-related properties. This fact makes probabilistic model checking a promising approach towards quantitative analysis of protocols [13, 25]. The importance of enabling quantitative analysis for a given cryptographic protocol was first shown in [26]. In that work, the author proposes a formal framework for weighting the cost of the participants, in order to verify the degree of a protocol's resistance against Denial of Service (DoS) attacks, in the context of the resource intensive task of mutual authentication. Recently, the aforementioned approach [26] has formed the basis for the analysis framework of [27]. The latter work [27] allows a more accurate representation of the protocol's computational cost. However, its drawback is that it employs simulation rather than verification for quantitatively analyzing the protocol's computational cost.

Another stochastic modeling approach for quantifying the availability of software systems under attack, is described in [28]. In this work the authors represent the system by a semi-Markov process (SMP). Using the appropriate SMP

model, they derive an embedded Discrete-Time Markov Chain (DTMC) [29], that consists of the necessary state transition probabilities. Once the steady-state DTMC probabilities have been computed, the sojourn time distributions for the model's states are used to compute the SMP's steady-state probabilities. This enables the calculation of the system's availability whilst concurrently, through parametric sensitivity analysis, the sensitivity of system's availability is studied. Stochastic modeling and analysis competence are required, since SMP is not developed by an automated analysis tool, such as PRISM. Furthermore, the system-level analysis does not take into account any resource expenditure for the considered states and, thus, it is not able to evaluate the protocol's messages processing cost.

Another quantitative analysis approach is described in [30]. The authors specify a three-way-handshake of the TCP protocol in probabilistic rewriting logic in order to verify DoS resistance. The described representation generates a model for the VESTA toolset [31]. It is basically a timed probabilistic model that is analyzed by Monte Carlo simulation, upon which a series of interrelated statistical hypothesis tests are applied, in order to check whether the quantitative property of interest is fulfilled. However, such an approach, also known as statistical model checking, does not produce the same accurate results as the ones obtained by probabilistic model checking [32]. Finally, the aforementioned approach is not appropriate for the analysis of communication protocols, since it does not cope with cost-related properties, such as the message generation and processing costs.

3. Protocol Analysis with Probabilistic Model Checking

PRISM is a powerful probabilistic model checking framework for verifying the quantitative properties of a system [13]. It supports three types of models: DTMCs (Discrete-Time Markov Chains), MDPs (Markov Decision Processes) and CTMCs. In this work, a CTMC representation of the CEMD protocol [14] is developed, together with an encoding of cost-related properties that are verified over the full state space of the model. CTMC analysis is chosen due to its strength in representing systems with dynamic behavior, e.g., communication protocols, and its advantage of being amenable to analytic treatment and numerical computation , e.g., calculation of computational and transmission cost over a finite time period. In the following paragraphs, the basic probabilistic model checking and CTMC principles, necessary for our work, are described.

3.1. Probabilistic Model Checking Principles

Model checking in general involves the verification of properties over labeled state transition systems [33]. In the context of probabilistic model checking, DTMC, CTMC and MDP are models that embed additional information about the likelihood of transitions between states to occur.

In PRISM, a probabilistic model is defined as a set of m modules (reactive modules), $M = \{M_1, \ldots, M_m\}$. Each M_i module is defined as a pair of

(Var_i, C_i), where Var_i is a set of integer-valued local variables with finite range and C_i is a set of commands. The set Var_i defines the local state space of module M_i and in turn Var denotes the set of all local variables in the model, i.e., $Var = \bigcup_{i=1}^{m} Var_i$. Furthermore, each variable $v \in Var$ has an initial value $\bar{v}$.

The behavior of module M_i is defined by the set of commands C_i. Each command $c \in C_i$ takes the form of $(g, (\lambda_1, u_1), \ldots, (\lambda_{n_c}, u_{n_c}))$, comprising a guard g and a set of pairs (λ_j, u_j), where $\lambda_j \in \Re_{>0}$ and u_j is an update for each $1 \leq j \leq n_c$. A guard g is a predicate over the set of all local variables Var and each update u_j corresponds to a possible transition of module M_i. If Var_i contains n_i local variables, $\{v_1, \ldots, v_{n_i}\}$, then an update takes the form $(v_1' = expr_1) \cap \ldots \cap (v_{n_i}' = expr_{n_i})$, where $expr_j$ is an expression in terms of the variables in Var. If an update leaves the values of some variables in Var_i unchanged, the model description may omit this information.

In CTMC model specification, the constants λ_j determine the rates attached to the transitions (i.e., the delay which occurs before the update takes place), whilst in DTMC model specification λ_j defines the probabilities attached to the transitions and, thus, $\lambda_j \in (0, 1]$ for $1 \leq j \leq n_c$ [29].

Example 1. The model of Listing 1 defines a CTMC consisting of a single module M_1. Module M_1 has a single local variable v with value range $\{1, \ldots, 5\}$ and initial value 1. Hence $Var = Var_1 = \{v\}$ and $\bar{v} = 1$. The following three lines describe the module's set of commands C_1. In the above notation each command $(g, (\lambda_1, u_1), \ldots, (\lambda_{n_c}, u_{n_c}))$ is written $[\,]g \rightarrow \lambda_1 : v_1 + \ldots + \lambda_{n_c} : v_{n_c}$. For example, the second command has the guard $(v > 1)\&(v < 5)$ and two updates, $v' = v - 1$ and $v' = v + 1$ on which a rate 0.5 is assigned. □

3.2. Continuous-Time Markov Chain (CTMC) principles

CTMCs are a commonly used technique in the field of performance and dependability analysis of computer and especially of real-time communication systems [34]. A CTMC is defined as a tuple $(S, \bar{s}, R, L)$, where:

- S is a finite set of states
- $\bar{s} \in S$ is the initial state
- $R : S \times S \rightarrow \Re_{\geq 0}$ is the transition rate matrix and
- $L : S \rightarrow 2^{AP}$ is the labeling function of atomic propositions AP that are true in S.

In CTMC the transition rate matrix R gives the rate of the transition, e.g., from a state s to s' within time t. For this transition, the actual probability will be $1 - e^{-R(s,s') \cdot t}$. Thus, in a CTMC model, the transition probability $P(s, s')$ in a single step will be:

$$P(s, s') = \begin{cases} \frac{R(s,s')}{\sum_{s'' \in S} R(s,s'')} & \text{if } s'' \neq s' \\ 1 & \text{if } s = s' \\ 0 & \text{if } s \neq s' \end{cases}$$

A path ω in a CTMC is a non-empty sequence $s_0 t_0 s_1 t_1 s_2 \ldots$, where, for all $i \geq 0$, there is a state $s_i \in S$ for which $R(s_i, s_{i+1}) > 0$ and $t_i \in \Re_{>0}$. The value t_i denotes the amount of time spent in the state s_i, while $\omega(k)$ represents the kth state of the path ω, i.e., s_k. A further detailed description of the CTMC semantics is provided in [17].

3.3. A brief description of Continuous Stochastic Logic (CSL)

CTMC properties are encoded as formulaes of the Continuous Stochastic Logic (CSL) [17]. In CSL the analyst develops partial specifications of the steady-state and the transient behavior of CTMCs. The syntax of CSL is as follows:

$$\phi \ ::= \ true \mid \alpha \mid \phi \ \wedge \ \phi \mid \neg\phi \mid P_{\bowtie p}[\psi] \mid \mathcal{S}_{\bowtie p}[\phi]$$

$$\psi \ ::= \ X \ \phi \mid \phi \, \mathcal{U}^{\leq t} \ \phi \mid \phi \, \mathcal{U} \ \phi$$

where a is an atomic proposition, operator $\bowtie \in [\leq, <, \geq, >]$, $p \in [0, 1]$ and $t \in \Re_{\geq 0}$. $P_{\bowtie p}[\psi]$ indicates that the probability of the path formula ψ, which is satisfied from a given state in a CTMC model, satisfies $\bowtie p$. CSL can also describe path formulaes with operators such as X (next) or $\mathcal{U}^{\leq t}$ (time bounded until with $t \in \Re_{\geq 0}$), as shown above. Finally, the $\mathcal{S}$ operator refers to the steady-state behavior of the CTMC. Formula $\mathcal{S}_{\bowtie p[\phi]}$ means that the steady-state probability of being in some state satisfying ϕ meets the bound $\bowtie p$.

As stated previously, the PRISM model checker also supports the specification and analysis of properties quantifying costs and rewards. This feature enables the analyst to use probabilistic model checking not only for measuring the actual probability of reaching a system state, but also, for determining a range of quantitative measures relating to the model's behavior.

For a CTMC $(S, \bar{s}, R, L)$, a reward structure is a tuple (ϱ, ι), where:

- $\varrho : S \to \Re_{\geq 0}$ is a vector of state rewards, and
- $\iota : S \times S \to \Re_{\geq 0}$ is a matrix of transition rewards.

A reward structure (ϱ, ι) for a CTMC allows the specification of four distinct types of rewards R:

- *Instantaneous* $R_{\bowtie r}[\mathrm{I}^{=t}]$: the expected value of the reward at time-instant t is $\bowtie r$,
- *Cumulative* $R_{\bowtie r}[\mathrm{C}^{\leq t}]$: the expected reward cumulated up to time-instant t is $\bowtie r$,
- *Reachability* $R_{\bowtie r}[\mathrm{F} \ \phi]$: the expected reward cumulated before reaching ϕ is $\bowtie r$,
- *Steady-state* $R_{\bowtie r}[\mathcal{S}]$: the long-run average expected reward is $\bowtie r$.

Cumulative reward properties are employed for the proposed quantitative verification of the proposed CTMC model.

4. Quantitative Analysis of the CEMD Protocol

4.1. The CEMD Protocol

E-mail protocols aim to provide strong fairness in order to ensure that the recipient receives the e-mail message, if and only if the sender receives the receipt of it. In [14] the authors propose such a protocol, namely the Fair Certified E-mail Delivery (CEMD). CEMD provides non-repudiation of origin, non-repudiation of receipt and strong fairness, whilst it makes use of an off-line and transparent trusted third party (TTP) only in exceptional circumstances, i.e., when the communicating parties fail to complete the e-mail for receipt exchange due to a network failure or a party's misbehavior. Besides the above, its design aimed at reducing the overall computational cost (i.e., cryptographic operations) for protocol's efficiency and cost-effectiveness [14].

More specifically, the CEMD protocol comprises two sub-protocols; the Exchange Protocol where the sender A and the recipient B attempt to exchange message M for its receipt, and the Receipt Recovery Protocol activated by the sender A when there is a need to send a request to TTP for receipt recovery after having failed to obtain B's receipt.

The Exchange Protocol of CEMD consists of the following four (4) discrete messages, also shown in Fig. 2:

1. $E_1 = (h(M), Sign_A(h(M)))$: sender A transfers to recipient B a hash value $h(M)$ of the message M along with its digital signature $Sign_A(h(M))$ on M.
2. $E_2 = (VRES_B, Cert_B)$: recipient B verifies $Sign_A(h(M))$ from message E_1 and confirms the result using $h(M)$. Then, he computes his Verifiable and Recoverable Encrypted Signature ($VRES_B$) message and sends to A a message E_2 consisting of $VRES_B$ and his certificate $Cert_B$, signed by TTP.
3. $E_3 = (M)$: message E_3 constitutes the e-mail message M that A wants to send to B.
4. $E_4 = (r_b)$: recipient B generates and sends a random prime number r_b to sender A in order A to be able to derive B's correct receipt using r_b.

The Receipt Recovery Protocol consists of three (3) discrete messages as depicted in Fig. 2:

1. $R_1 = (M, Cert_B, VRES_A)$: the sender A transfers to the TTP the message R_1 consisting of the message M, the recipient's B certificate $Cert_B$ and its $VRES_A$, where $VRES_A$ is derived from $VRES_B$ such that TTP accepts A's request.
2. $R_2 = (r_b)$: Once the sender A receives R_1, TTP confirms that sender A has sent the correct message M in order to avoid the unlawfully recover of B's receipt r_b for him. In other words, $h(M)$, received by party B in step E_1 and used for computing $VRES_B$, must be identical to the hash value of message M computed by TTP in the first step of the Receipt Recovery Protocol. As a result, TTP will decrypt $VRES_A$ using its private key in

order to derive a secret key between TTP and party B and using it to recover receipt r_b. Finally, TTP sends to A the message R_2 containing the r_b in order that A computes the receipt of the recipient B.

3. $R_3 = (M)$: TTP forwards the original message M to the recipient B.

CEMD protocol has to provide certain security properties towards its participants. According to [14, 15], the CTMC model of CEMD protocol is designed in respect to the following characteristics:

- *fairness*, i.e., neither the sender nor the recipients should they gain advantage in a random protocol's interruption. For this reason CEMD participants are considered to employ an off-line TTP entity to ensure fair completion of the CEMD protocol.

- *timeliness*, i.e., all of the participants should be able to successfully terminate the protocol in a given finite time. We considered that all sessions once initiated will not fail. This design feature will force successful completion of each session in a finite amount of time allowing us to anticipate timeliness property in results.

- *confidentiality*, i.e., only the intended recipient should learn the contents of an e-mail message. Confidentiality of the participants is kept, since RSA encryption (using modulo exponentiations), of all CEMD messages is considered unbreakable.

- *TTP invisibility*, i.e., the TTP should only be used in cases of network failure or participants' misbehavior. In the proposed model the TTP entity involves in the CEMD session only when participant A fails to receive a proper receipt of B. Otherwise TTP does not interfere to the protocol resulting in TTP invisibility.

Moreover, the developed CTMC model supports not only two participants, namely sender A and recipient B, but a number of multiple sessions that sender A keeps simultaneously with a set of recipients $B = \{B_1, \ldots, B_n\}$, according to Table 1, which represents a more realistic scenario. However, assurance of these properties entails strong cryptographic operations [15], and thus, increased computational cost for the protocol's participants. It is therefore questionable whether a secure protocol, such as the one described, can effectively operate over environments characterized by requirements for low-cost hardware, such as mobile devices. Furthermore, it is fundamental for a secure protocol designed for mobile environments to be tested in line with the feature of BER, since BER determines the transmission cost, which along with the computational cost imposes considerable restrictions.

Thus, the purpose of the proposed analysis is to provide quantitative results for the cost-related properties of the CEMD protocol, i.e., computational and transmission cost. These results concern multiple sessions among sender A and the set of recipients B that are supposed to take place in mobile environments.

The value of this analysis is that it provides answers in whether or not a communication protocol is a cost-permitted approach in environments with specific requirements such as mobile environments.

For the above purpose, we firstly built a CTMC model for the CEMD protocol [14]. The Markov Chain created was augmented with cumulative reward properties R that represent the computational and transmission cost imposed when the model reaches a state, i.e., s. Then, the CEMD protocol was executed by configuring appropriately the BER and considering up to 7 parallel sessions. The visibility of the TTP entity was also considered as an execution parameter.

4.2. Probabilistic Model Checking of CEMD

The proposed CTMC analysis of the CEMD protocol (CEMD-CTMC) consists of four (4) procedural steps, namely the DEVELOP, DERIVE, ADOPT and EMBED steps, as shown in Fig. 3.

4.2.1. DEVELOP *the CEMD-CTMC Model*

The CEMD-CTMC model, developed in the PRISM model checker, comprises three (3) modules, namely $M = \{M_A, M_B, M_{TTP}\}$, corresponding to the entities shown in Table 1. Module M_A represents the sender A and initiator of the e-mail exchange protocol, while module M_B embodies a fixed set of recipients B communicating with A, thus modeling a parallel sessions scenario. It is assumed that all the recipients B_i, $i = 1, \ldots, 7$ trust the same TTP entity, represented by the module M_{TTP}.

The structure of the CTMC model supports probabilistic model checking for up to $n = 7$ parallel sessions, one for each of B_i recipient, $i = 1, \ldots, 7$, in order to emulate a real-world scenario of e-mail delivery. Nowadays, it is usual that a simple email user initiates a fair certified email session with more than one participants concurrently, but at the same time hardware limitations pose restrictions to a limited number of parallel sessions. Considering parallel sessions, the modules of set M interact by updating their local variables in line with the protocol's communication steps presented in Section 4.1. These updates over a finite set of variables correspond to the modeled state transitions for the distinct participants. The number of recipients is defined by the constant $number\_of\_B\_machines$, as shown in Listing 2. TTP can either be visible or not during a session according to A's choice, as prescribed by the CEMD principles [14]. Table 2 provides the notations used in the PRISM model.

Two parameters associated with the cost-related properties of CEMD are defined, namely the computational and the transmission cost. More specifically, parameter ver_E_i, $i = 1, \ldots, 4$, is associated with the computational cost, since it defines the rate of positive verification of a message E_i, $i = 1, \ldots, 4$. The rate of negative verification of E_i message is defined as $1 - ver\_E_i$ and it is associated with the transmission cost, since it results in message retransmission. Parameter ber is also related with the transmission cost, since it denotes the rate for an error to be occurred during the transmission of a message E_i, $i = 1, \ldots, 4$. The rate of the successfully transmission of a message E_i is defined as $1 - ber$. In Listing 2, it

is assumed that the verification rate remains the same for verifications performed by the same entity. Thus, $ver\_E_1 = ver\_E_3$ and $ver\_E_2 = ver\_E_4$, since the verification of E_1 and E_3 messages is performed by recipients B_i, $i = 1, \ldots, 7$, while messages E_2 and E_4 are verified by the sender A.

Parameters $TTP\_certificates$ and no_TTP are associated with the TTP invisibility security property. More specifically, $TTP\_certificates$ is defined in M_A and counts the number of certificates that TTP produces. This entails that $TTP\_certificates$ counts simultaneously the sessions completed with TTP being visible. On the other hand, no_TTP is a counter, also defined in M_A, for sessions successfully completed with TTP being invisible. Hence, the formula $finish$ that is supposed to be satisfied in the model's final state, is defined as the sum of $TTP\_certificates$ and no_TTP counters that has to be equal to $number\_of\_B\_machines$.

For all modules developed in our model, namely M_A, M_B, M_{TTP}, a number of local control variables is fixed and defined to have a maximum value equal to $number\_of\_B\_machines$, which represents the number of parallel sessions considered in the proposed analysis. For example, $ACK\_counter\_A$ local control variable of M_A counts the initial ACK messages received by recipients B, as shown in Listing 3. Thus, $ACK\_counter\_A \in Var_A$ is used in guard $g = (ACK\_counter\_A < number\_of\_B\_machines)$ which controls the commands c_1 and c_2 shown in lines 7 and 9 of Listing 3, with $c_1, c_2 \in C_A$. If g is TRUE command c_1 will perform the update $u_1 = (ACK\_counter\_A' = ACK\_counter\_A + 1)$ & $(A\_state' = 1)$ with rate $\lambda_1 = 1 - ber$ and respectively c_2 command will perform the update $u_2 = (A\_state' = 0)$ with rate $\lambda_2 = ber$. For the synchronization of M_A and M_B synchronization labels are used, e.g., $[A\_send\_ACK]$ in line 7 of M_A and 18 of M_B, presented in Listing 3. The interpretation of the code in Listing 3 is that A will send a maximum of ACK messages according to the $number\_of\_B\_machines$ parameter each of which is transmitted successfully with a rate of $1-ber$ in accordance with BER in mobile environments.

4.2.2. Derive *Computational and Transmission Cost Parameters*

In this step, the computational and transmission cost parameters of the CEMD protocol are derived from the CEMD specifications [14]. More specifically, computational cost is calculated in line with the CPU cycles required for processing CEMD messages, i.e., $E_1 - E_4$ and $R_1 - R_3$, as it is stated in Section 1.1. Table 3 presents the CPU cycles consumed in encryption and decryption actions of messages $E_1 - E_4$ and R_1, R_2 verification, when the widely used Texas Instruments TMS320C55x Family operating at $200MHz$ is employed [19, 20] . More specifically, in the analysis the encryption and decryption cost are calculated at 8.84×10^6 and 36.36×10^6 CPU cycles, respectively, when the corresponding action takes place at the sender A or the recipients B_i, $i = 1, \ldots, 7$. However, the encryption process during the protocol's step R_1 is executed at the TTP, which is considered to operate at $2GHz$, and thus the CPU cycles are computed at 8.84×10^5. The computational cost of other mathematical calculations, such as multiplication or division, is considered to

be negligible. Transmission cost is a function of negative verifications and BER occurrence, since both cases entail message retransmission. Obviously, cost calculation considers the parallel sessions scenario, the CEMD-CTMC model is based on, and TTP visibility, since transmission of only $E_1 - E_4$ messages entails TTP invisibility, whilst $E_1 - E_4$ followed by $R_1 - R_3$ indicate TTP visibility.

Based on Table 3, the reward modules of the CEMD-CTMC model, described in Section 4.2.4, are constructed.

4.2.3. ADOPT *Bit-Error-Rate Parameter*

According to Section 4.2.2 BER is one of the two parameters that cause the model's transmission cost. The values of BER in mobile communications are considerable high compared to those in wired networks and, thus, they should be taken into account. Even in cases where protocols such as HSDPA are adopted, the values of BER vary from 10^{-6} to 10^{-3} [10, 11, 35, 36]. The proposed CEMD analysis considers this range of BER values as shown in Table 2. The presence of BER in mobile networks constitutes them as noisy environments, since CEMD messages may be delivered with errors, and thus, their retransmission is required. As an example, Listing 4 shows sender A and recipient B_i, $i = 1, \ldots, 7$, exchanging message E_1.

Synchronization label $[A\_send\_E1\_error]$ in line 5 of M_A and 16 of M_B indicates that message E_1 has been successfully transmitted by A with a rate $1 - ber$ and received by B_i. Thus, the state of A is updated, i.e., $(A\_state' = 3)$. In case of error occurrence, A returns to $(A\_state' = 2)$ with rate ber. Upon receipt of E_1, B_i proceeds to message's verification. Positive verification occurs with rate ver_E_1 and indicates an update in B_i' state, i.e., $(B\_state' = 4)$, while in case of negative verification the modules are synchronized through the label $[incorrect\_ver\_of\_E_1]$. This occurs with rate $1 - ver\_E_1$ and entails that both participants go back to their previous states, i.e., $(A\_state' = 2)$ and $(B\_state' = 2)$.

Listing 4 along with Table 3 makes clear the way transmission cost is incorporated in the proposed analysis taking into account both BER occurrence and negative verification rates. Apparently, the same logic for the transmission cost calculation is employed in protocol's steps (E_2, E_3, E_4 and R_1, R_2, R_3) that follow.

4.2.4. EMBED *cost parameters in rewards*

At this final step, the cost parameters, described in Section 4.2.2, as well as the *ber* parameter, described in Section 4.2.3, are embedded in the proposed CEMD-CTMC model as rewards. Thus, in accordance with Table 3, rewards for both the computational and transmission cost of CEMD should be defined.

More specifically, "Exponentiation" rewards count the CPU cycles required in messages' verification due to encryption and decryption processes. For example, in Listing 4 the transition labeled $[verification\_of\_E_1]$ (line 18) triggers an increase of computational cost. "Retransmission" rewards measure messages' transmission cost based on the errors occurred during a message's transmission

as they are expressed by BER parameter. For example, in Listing 4 the transitions labeled [$A\_send\_E_1\_error$] (line 7) and [$incorrect\_ver\_of\_E_1$] (line 20) trigger an increase of transmission cost.

5. Results

As it has been mentioned, the proposed analysis aims at providing quantitative results for the cost-related properties of the CEMD protocol, i.e., computational and transmission cost. These results consider multiple sessions among sender A and a set of recipients $B = \{B_1, \ldots, B_n\}$, $n = 7$. The aforementioned entities are considered to belong to the widely used Texas Instruments TMS320C55x Family and operate at $200MHz$ [19, 20] inside a HSDPA mobile environment, where BER varies from 10^{-6} to 10^{-3} [10, 11, 35, 36].

The developed CTMC-CEMD model preserves the security characteristics delivered by the CEMD protocol. CEMD is designed in this way as to guarantee security properties such as fairness and confidentiality for the involved participants. The authors in [14] propose the use of the RSA cryptosystem for the exchange messages that will preserve fairness of the participants. In this way, even if the protocol fails in one of its steps, sender A or recipient B_i, $i = 1, \ldots, 7$, will not gain additional knowledge against each other. Moreover, in the developed model there is no intruder entity involved in the communication between the CEMD participants. Such an assumption offers, by default, confidentiality for all of the protocol participants.

For the quantitative analysis of the proposed CEMD-CTMC model, cumulative reward properties of the form $R_{\bowtie r}[\mathrm{C}^{\leq t}]$, are employed as mentioned in Section 3.3. Cumulative reward properties associate a reward with each path of the model, but only up to a given time bound. The property $[\mathrm{C}^{\leq t}]$ corresponds to the reward cumulated along a path until t time units have elapsed. Timeliness, as a security property that an e-mail protocol should provide, can be verified through a user defined time boundary. Thus, the first CSL query Q_1 is defined as a cumulative type query as follows:

$$Q_1 : R\{\text{“}Computational\_Cost\text{”}\} =? \ [C <= C_0]$$

whose interpretation is: “which is the overall $Computational\_Cost$ for completing all protocol's sessions in a finite amount of time C_0?”. Fig. 4 provides results for CEMD's computational cost as a function of time (expressed in time units) for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$.

More specifically, in the Q_1 structure $R\{\text{“}Computational\_Cost\text{”}\}$ composes the reward structure (ϱ, ι) which is defined inside the CEMD-CTMC model according to the cost parameters of CEMD specifications shown in Table 3 (column CPU cycles). C_0 is a constant representing the time boundary t and varies from 0 to 75 with step 2, as shown in Fig. 4.

Apart from timeliness and successful completion of protocol, TTP invisibility should also be studied. Given that TTP ought to be used in cases of network failures or participant's misbehavior, it would be useful to study the

CEMD's computational cost with respect to TTP invisibility scenarios. Thus, for the analysis purpose, three different cases and the results are shown in Fig. 4. Firstly, in Fig. 4(a), the probability of TTP being visible is defined $P\_TTP \simeq 1$ indicating that TTP will be involved in all participants' sessions, secondly, in Fig. 4(b) the TTP's visibility is selected with $P\_TTP = 0.5$ and finally, in Fig. 4(c), the CEMD protocol completes its sessions with TTP being invisible and, thus, $P\_TTP \simeq 0$.

As it is expected the higher the probability of TTP visibility the greater the computational cost of the protocol. This is confirmed by the y-axis of Fig. 4(a)-Fig. 4(c) which depicts the number of CPU Mcycles required by the corresponding mobile devices operating at $200MHz$ to complete the CEMD protocol. As it is mentioned in Section 4.2.2, these results are derived based on the calculation that the encryption and decryption cost are 8.84×10^6 and 36.36×10^6 CPU cycles, respectively, when the corresponding action takes place at the sender A or the recipients B_i, $i = 1, \ldots, 7$. When the encryption process is executed by the TTP, i.e., during the protocol's step R_1, then the CPU cycles are 8.84×10^5 CPU cycles, since TTP is considered to operate at $2GHz$. At the same time, as the probability of TTP being visible increases, the overall protocol completion time increases as well. For example, for $n = 4$ recipients, i.e., $B = \{B_1, B_2, B_3, B_4\}$, the protocol's computational cost for $P\_TTP \simeq 0$ (Fig. 4(c)) is 1009 CPU Mcycles and all the sessions require about 54 time units to be completed. When $P\_TTP = 0.5$ (Fig. 4(b)) the respective cost is 1044 CPU Mcycles and time rises to 56 units and, finally, for $P\_TTP \simeq 1$ (Fig. 4(a)) the cost is 1080 CPU Mcycles and time units are 58. The interpretation for the above observations lies on the definition of the computational cost, which is a straight function of the CPU cycles required for verifying $E_1 - E_4$ and R_1, R_2 messages. Since messages R_1 and R_2 are associated with the TTP visibility, it is obvious the probability of TTP's visibility has a positive relationship with the overall computational cost, as well as with the amount of time for the successful completion of the protocol's session.

A second observation is that the more the protocol's recipients the greater the computational cost of the protocol, as indicating by the level of each curve in Fig. 4(a)-Fig. 4(c). Obviously, the same stands for the relationship between the number of recipients and the protocol's completion time. Indicatively, when $P\_TTP \simeq 0$ (Fig. 4(c)), for $n = 2$, i.e., $B = \{B_1, B_2\}$, the protocol's computational cost is 504 CPU Mcycles and all the sessions require about 40 time units to be completed. These observations are lower than the ones for $n = 4$. For $P\_TTP = 0.5$ (Fig. 4(b)) the cost for $n = 2$ recipients is 522 CPU Mcycles and time rises to 44 units, while for $P\_TTP \simeq 1$ (Fig. 4(a)) the cost is 540 CPU Mcycles and time units are 48. The explanation in this case is that more protocol's recipients entail more messages to be exchanged and thus more exponentiations to be computed.

Then, two separate scenarios are launched for transmission cost analysis. In the first one we study the impact of varying the verification rate of sender A, while in the second scenario the verification rate of the recipients $\{B_1, \ldots, B_7\}$ is changed.

More specifically, the cumulative query Q_2 is defined in order to obtain the protocol's transmission cost under the assumption that the verification rate of sender A varies:

$$Q_2 : R\{\text{“}Transmition\_Cost\text{”}\} =? \ [C <= C_0], \text{ and } a = 0.7, 0.8, 0.9$$

The interpretation of this query is: “which is the overall $Transmission\_Cost$ for completing all protocol's sessions in a finite amount of time C_0 for different verification rates of sender A?”. The answer is graphically presented in Fig. 5, which depicts the CEMD's transmission cost as a function of time (expressed in time units) for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, and for $a = 0.7, 0.8, 0.9$ verification rates of sender A. The verification rates of sender A are defined to be close to 1, since the entity A initiates the protocol and thus it is considered to be more reliable to complete it successfully.

The above values of a have been chosen to be at a high rate level, since the entity A is the protocol's initiator and, thus, it is considered to have a low probability for negative verifications (e.g., it initiates a session only when it has the required bandwidth). Fig. 5(a)-5(c) show the protocol's transmission cost for 1 up to 7 parallel sessions when $ber = 10^{-3}$ and for $a = 0.7, 0.8, 0.9$, respectively. The effect of A's verification rate is that the higher the value of a the lower the protocol's cost. This is because positive verifications reduce retransmissions. Indicatively, for $n = 4$ recipients, i.e., $B = \{B_1, B_2, B_3, B_4\}$, the protocol's transmission cost for $a = 0.7$ (Fig. 5(a)) is 2.95 and all sessions require about 42 time units to be completed. These values are lower than the respective cost of 1.84 and time of 34 units when $a = 0.8$ (Fig. 5(b)). Finally, for $a = 0.9$ (Fig. 5(c)) the cost is further decreased at 0.88 and time units drop to 32.

Similarly to the results derived by query Q_1, the increase in the number of recipients entails greater transmission cost, while timeliness property is preserved. This is also natural in case of query Q_2, since more recipients B_i, $i = 1, \ldots, 7$, trigger the exchange of more messages. More messages not only increase the transmission cost but also cause delay in sessions completion. Thus, all curves in each sub-figure “move up” in line with the increase in the number of recipients. At the same time, curves in Fig. 5(a) are “shifted” to the right compared those in Fig. 5(b), which are “shifted” to the right compared to those in Fig. 5(c).

The next cumulative query Q_3 also concerns the protocol's transmission cost but under different verification rates of recipients of set B:

$$Q_3 : R\{\text{“}Transmition\_Cost\text{”}\} =? \ [C <= C_0], \text{ and } b = 0.3, 0.5, 0.7$$

The interpretation of Q_3 is: “which is the overall $Transmission\_Cost$ for completing all protocol's sessions in a finite amount of time C_0 for different verification rates of all the recipients of set B?”. Fig. 6 provides the answer, since it presents CEMD's transmission cost as a function of time (expressed in time units) for $\{B_1, \ldots, B_7\}$ recipients.

In this scenario, low rates of b correspond to high probability of negative verifications, since recipients B_i, $i = 1, \ldots, 7$, are the protocol's responders

(e.g., they may be prone to errors due to hardware limitations or device misbehavior). Fig. 6(a)-6(c) depicts the protocol's aggregated transmission cost for $b = 0.3, 0.5, 0.7$, respectively, and for 1 up to 7 parallel sessions when $ber = 10^{-3}$. The effect of Bs' verification rate is that the higher the value of b the lower the protocol's cost and the earlier the time for sessions' completion. This is because positive verifications reduce retransmissions. Moreover, in all three sub-figures, the number n of recipients has a straight positive relationship with the protocol's transmission cost, while it also affects the overall time required for successful completion of the CEMD sessions.

In practice, we observe that for $n = 4$ parallel sessions, if verification rate of recipients is $b = 0.3$ (Fig. 6(a)) the transmission cost reaches 18.72 after 88 time units. If verification rate is $b = 0.5$ both cost and time for completion are lower, i.e., 8.04 and 56 time units, respectively, and finally for $b = 0.7$ the corresponding values are much lower at 3.46 and 40 time units. Obviously, if parallel sessions increase the above values will also be increased keeping their relationship.

Quantitative analysis results derived from Fig. 4-6 show that all curves are "fixed" after a finite time period indicating that cost is also "fixed", since no more protocols actions either processing or transmission take place (however, the higher verification rates of sender A lead to earlier convergence compared to the lower verification rates of recipients which result in convergence delay, as indicating by the x-axis of Fig. 5 and Fig. 6). This behavior is caused because our model has been designed not to fail even if a number of faults occur (e.g., $[A\_send\_E_1\_error]$, $[incorrect\_ver\_of\_E_1]$ in Listing 4). Such a design feature will eventually allow the successful e-mail message delivery for all the initiated CEMD sessions.

Time convergence, presented in Fig. 4-6, is also confirmed when different values of *ber* are employed. The results of Tables 4 and 5 refer to transmission cost and they are derived when the indicative in mobile environments values of $ber = 10^{-6}$ and $ber = 10^{-3}$ are used [35, 10, 11, 36]. In Table 4 the different values of BER are studied under $a = 0.8$ verification rate of sender A, whilst in Table 5 we set the verification rate of recipients, i.e., $\{B_1, \ldots, B_7\}$ where $n = 1, \ldots, 7$, at $b = 0.5$. As it is expected higher values of *ber* result in increasing transmission cost. In fact, the more the parallel sessions the bigger cost overhead. Indicatively, for $n = 7$ the cost overhead reaches at 4.8% after 60 time units when $a = 0.8$ and at 5.7% after 80 time units when $b = 0.5$. The higher values of Table 5 compared to those of Table 4 as well as the later time convergence are due to the low verification rate of the recipients.

An extra confirmation of the above behavior is provided by the results of query Q_4 that follows. We define the PCTL query Q_4:

$$Q_4 : P =? \, [F <= C_0 \; finish], \text{ and } b = 0.2, 0.9$$

where the argument $finish$ is a boolean formula defined in our model (Listing 3, line 18) to check whether or not the initiated sessions will be completed successfully. The interpretation of Q_4 is: "which will be the computed probability of all CEMD sessions being completed successfully in the time boundary

C_0 for different verification rates of all the recipients of set B?". The answer is provided by the Fig. 7, which shows the probability of $B_1, \ldots, B_7$ recipients completing their sessions as a function of time.

More specifically, Fig. 7(a) and Fig. 7(b) depict the probability of all CEMD sessions being completed when the verification rate of all the recipients of set B is $b = 0.2$ and $b = 0.9$, respectively. These values correspond to a representative range for recipients' verification rate. The different curves in each figure correspond to the different number of recipients, from $n = 1$ up to $n = 7$. Obviously, all the curves, in both sub-figures, converge at 1, indicating the successful completion of all protocol sessions. However, time differentiates these curves, i.e., the more the protocol's recipients the later the time of curves' convergence. Moreover, the higher the verification rate of recipients the earlier the convergence, as indicating by the x-axis of Fig. 7(a) and Fig. 7(b). A low verification rate for recipients B_i, i.e., $b = 0.2$, entails an unstable environment, since it means that recipients succeed in verifying exchanging messages with a low probability. But, a low probability leads to failures and messages' retransmission, thus, in delays in sessions' completion. For example, in Fig. 7(a) ($b = 0.2$), we observe that the probability of completing successfully the protocol becomes 1 after 147 time units for $n = 7$ and almost 75 time units for $n = 1$. On the other hand, for a higher verification rate, $b = 0.9$ (Fig. 7(b)), all the sessions are completed with probability 1 after a period of 40 time units when $n = 7$ and after 25 time units when $n = 1$. Comparing the two graphs we can also observe that, for example, when $n = 3$, the probability for successful completion of all the three sessions will be 1 in 30 time units for $b = 0.9$, while for $b = 0.2$ in 30 time units the probability of all three sessions being completed is 0.13.

Such divergences indicate that the CEMD protocol is prone to the specific hardware characteristics of participants, since, for example, an obsolete or low-cost device produces delays in executing exponentiations. At the same time, the CEMD protocol is prone to the environment it operates, since high BER leads to transmissions errors, which result in negative verifications and retransmissions.

6. Conclusions and a Look Ahead

This work proposes a quantitative analysis as a means for the evaluation of the CEMD protocol operating in prone to errors environments, such as mobile networks, while supporting parallel sessions. Our analysis is based on the development of a parameterized CTMC model which implements parallel CEMD sessions. The CEMD-CTMC model is designed in respect to the security properties imposed by the CEMD protocol and, then, is augmented with specific computational and transmission cost parameters derived from CEMD specifications. The probabilistic model checking analysis, launched, aims at quantitatively verifying the general cost of protocol. Such an approach gives insights for deciding on whether the CEMD protocol can operate upon hardware devices with specific power limitations (e.g., computational resources) and under environments with specific BERs.

As a future research prospect, we plan to extend our model in order to tune computational rewards according to hardware specifications of devices used in mobile communications (e.g., CPU processing power, memory capabilities) and transmission rewards according to specific networks' bandwidth. A second insight is to study computational cost in line with energy consumption (e.g., battery life). Finally, it would be interesting to build a general quantitative probabilistic model checking framework which will be a tool-assistant, protocol-independent approach for analysts/designers, in order to verify their properties of interest.

References

[1] G. Lowe, Breaking and fixing the needham-schroeder public-key protocol using fdr, Software - Concepts and Tools 17 (3) (1996) 93–102.

[2] G. Lowe, A. W. Roscoe, Using csp to detect errors in the tmn protocol, IEEE Transactions on Software Engineering 23 (10) (1997) 659–669.

[3] S. Basagiannis, P. Katsaros, A. Pombortsis, N. Alexiou, A probabilistic attacker model for quantitative verification of dos security threats, in: Proc. of the 32nd Annual IEEE International Conference on Computer Software and Applications (COMPSAC'08), Finland, 2008, pp. 12–19.

[4] D. Miorandi, E. Uhlemann, S. Vitturi, A. Willig, Guest editorial: Special section on wireless technologies in factory and industrial automation, part i, IEEE Transactions on Industrial Informatics 3 (2) (2007) 95–98.

[5] Q. Bi, G. Zysman, H. Menkes, Wireless mobile communications at the start of the 21st century, IEEE Communications Magazine 39 (1) (2001) 110–116.

[6] Y. Zhang, N. Ansari, H. Tsunoda, Wireless telemedicine services over integrated ieee 802.11/wlan and ieee 802.16/wimax networks, IEEE Communications Magazine 17 (1) (2010) 30–36.

[7] C. Liaskos, S. Petridou, G. Papadimitriou, Cost-aware wireless data broadcasting, IEEE Transactions on Broadcasting 56 (1) (2010) 66–76.

[8] N. Sklavos, K. Touliou, A system-level analysis of power consumption & optimizations in 3g mobile devices, in: Proc. of the 1st International Conference on New Technologies, Mobility & Security (NTMS'07), France, 2007, p. 225235.

[9] M. Mouly, M. Paulet, The GSM System for Mobile Communications, published by the authors, ISBN 2-9507190-0-7, 1992.

[10] N. K. C. K. Sohaib, J. Nordberg, Hsdpa system simulation, in: Proc. of the 2nd International Symposium on Communications, Control and Signal Processing (ISCCSP'06), Morocco, 2006.

[11] D. Kliazovich, M. Devetsikiotis, F. Granelli, Formal methods in cross layer modeling and optimization of wireless networks: State of the art and future directions, in: S. Kotsopoulos, K. Ioannou (Eds.), Heterogeneous Next Generation Networking: Innovations and Platforms, IDEA Group Inc., 2008, pp. 1–24.

[12] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[13] M. Z. Kwiatkowska, G. Norman, D. Parker, Prism 2.0: A tool for probabilistic model checking, in: Proc. of the 1st Int. Conf. on Quantitative Evaluation of Systems (QEST'04), Netherlands, 2004, pp. 322–323.

[14] A. Nenadic, N. Zhang, S. Barton, Fair certified e-mail delivery, in: Proc. of the 2004 ACM Symposium on Applied Computing (SAC'04), USA, 2004, pp. 391–396.

[15] G. Ateniese, C. Nita-Rotaru, Stateless-recipient certified e-mail system based on verifiable encryption, in: Proc. of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology (CT-RSA'02), Germany, 2002, pp. 182–199.

[16] W. J. Stewart, Introduction to the numerical solution of Markov chains, Princeton University Press, 1994.

[17] A. Aziz, K. Sanwal, V. Singhal, R. K. Brayton, Model-checking continous-time markov chains, ACM Transactions on Computational Logic 1 (1) (2000) 162–170.

[18] Prism model checker, World Wide Web electronic publication.
URL `http://www.prismmodelchecker.org/`

[19] R. Hwang, F. Su, L. Huang, Fast firmware implementation of rsa-like security protocol for mobile devices, Wireless Personal Communications 42 (2007) 213–223.

[20] R. Hwang, F. Su, L.Huang, J. Peng, Implementing the rsa algorithm on the ti tms320c55x family, in: Proc. of the 37th IEEE International Carnahan Conference on Security Technology, USA, 2003, pp. 569–572.

[21] P. Prasithsangaree, P. Krishnamurthy, On a framework for energy-efficient security protocols in wireless networks, Computer Communications 27 (17) (2004) 1716–1729.

[22] R. Mayo, P. Ranganathan, Energy consumption in mobile devices: Why future systems need requirements-aware energy scale-down, Special Issue on Power Management (2003) 26–40.

[23] R. Karri, P. Mishra, Minimizing energy consumption of secure wireless session with qos constraints, in: Proc. of the 8th International Conference on Communications (ICC'08), USA, 2002, pp. 2053–2057.

[24] P. Katsaros, A roadmap to electronic payment transaction guarantees and a colored petri net model checking approach, Information & Software Technology 51 (2) (2009) 235–257.

[25] S. Basagiannis, P. Katsaros, A. Pombortsis, N. Alexiou, Probabilistic model checking for the quantification of dos security threats, Computers & Security 28 (6) (2009) 450–465.

[26] C. Meadows, A cost-based framework for analysis of denial of service in networks, Journal of Computer Security 9 (1-2) (2001) 143–164.

[27] S. Tritilanunt, C. Boyd, E. Foo, J. M. G. Neto, Using coloured petri nets to simulate dos-resistant protocols, in: Proc. 7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Denmark, 2006, pp. 261–280.

[28] B. B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, K. S. Trivedi, Modeling and quantification of security attributes of software systems, in: Proc. of the IEEE/IFIP Int. Conference on Dependable Systems and Networks (DSN'02), Washington, DC, USA, 2002, pp. 505–514.

[29] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: Proc. of the 15th Conference on Foundations of Computer Technology and Theoretical Computer Science, India, 1995, pp. 499–513.

[30] G. Agha, M. Greenwald, C. A. Gunter, S. Khanna, J. Meseguer, K. Sen, P. Thati, Formal modeling and analysis of dos using probabilistic rewrite theories, in: Proc. the IEEE Workshop on Foundations of Computer Security (FCS'05), Chicago, IL, USA, 2005.

[31] K. Sen, M. Viswanathan, G. Agha, Vesta: A statistical model-checker and analyzer for probabilistic systems, in: Int. Conference on Quantitative Evaluation of Systems, Los Alamitos, CA, USA.

[32] K. Sen, M. Viswanathan, G. Agha, On statistical model checking of stochastic systems, in: Proc. of the 17th Int. Conference on Computer Aided Verification (CAV'05), UK, 2005, pp. 266–280.

[33] E. Clarke, O. Grumberg, D. Peled, Model Checking, The MIT Press, 2000.

[34] M. Kwiatkowska, G. Norman, D. Parker, J. Sproston, Modeling and Verification of Real-Time Systems: Formalisms and Software Tools, John Wiley & Sons, 2008.

[35] G. Varrall, R. Belcher, 3G Handset and Network Design, Wiley, 2003.

[36] P. Nicopolitidis, M. Obaidat, G. Papadimitriou, A. Pomportsis, Wireless Networks, John Wiley & Sons, 2003.

Table 1: CTMC-CEMD Entities

Table 2: CTMC-CEMD parameters' notation

Table 3: Computational and Transmission Cost Parameters

Table 4: Transmission cost as a function of time for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, $a = 0.8$ verification rate of sender A and $ber = 10^{-6}, 10^{-3}$

Table 5: Transmission cost as a function of time for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, whose verification rate is $b = 0.5$ and $ber = 10^{-6}, 10^{-3}$

Listing 1: Example of a CTMC model in PRISM language

Listing 2: Definition of CTMC-CEMD parameters

Listing 3: Control variables and modules' synchronization

Listing 4: Synchronization of A and B_i during the exchange of E_1

Figure 1: An abstract representation of the proposed analysis

Figure 2: An overview of the CEMD protocol

Figure 3: The CEMD quantitative analysis using probabilistic model checking

(a) Probability of TTP's visibility $P_{TTP} \simeq 1$

(b) Probability of TTP's visibility $P_{TTP} = 0.5$

(c) Probability of TTP's visibility $P_{TTP} \simeq 0$

Figure 4: Computational cost as a function of time for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$

(a) $a = 0.7$

(b) $a = 0.8$

(c) $a = 0.9$

Figure 5: Transmission cost as a function of time for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, and for different verification rates of sender A: $a = 0.7, 0.8, 0.9$

(a) $b = 0.3$

(b) $b = 0.5$

(c) $b = 0.7$

Figure 6: Transmission cost as a function of time for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, whose verification rates are: $b = 0.3, 0.5, 0.7$

(a) $b = 0.2$

(b) $b = 0.9$

Figure 7: The probability for $n = 1, \ldots, 7$ recipients, i.e., $\{B_1, \ldots, B_7\}$, to complete their sessions as a function of time

Symbol	Description
A	Protocol's sender
$B = \{B_1, \ldots, B_n\}$	Set of recipients communicating with A, $n = 7$
TTP	Trusted Third Party

Symbol	Description
$number\_of\_B\_machines$	Number of recipients $n = 7$
ber	Bit error rate ($10^{-6} - 10^{-3}$)
ver_E_i	Rate of positive verification for message E_i, where $i = [1..4]$
$TTP\_certificates$	Certificates produced by TTP and received by A
no_TTP	Sessions completed with TTP being invisible
$finish$	Formula representing the final state of the model

Protocol Step	CPU cycles	Negative Verification	BER occurrence
E_1	62.88×10^6	✓	✓
E_2	54.04×10^6	✓	✓
E_3	8.84×10^6	✓	✓
E_4	36.36×10^6	✓	✓
R_1	8.84×10^5	-	✓
R_2	8.84×10^6	-	✓
R_3	-	-	✓

Time	14		28		44		60		74	
ber	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}
$n = 1$	0.418	0.424	0.451	0.458	0.452	0.458	0.452	0.458	0.452	0.458
$n = 2$	0.753	0.765	0.903	0.917	0.904	0.917	0.904	0.917	0.904	0.917
$n = 3$	0.988	1.005	1.353	1.374	1.356	1.376	1.356	1.376	1.356	1.376
$n = 4$	1.135	1.156	1.799	1.826	1.808	1.835	1.808	1.835	1.808	1.835
$n = 5$	1.216	1.240	2.233	2.267	2.260	2.294	2.260	2.294	2.260	2.294
$n = 6$	1.252	1.279	2.644	2.684	2.712	2.753	2.712	2.753	2.712	2.753
$n = 7$	1.265	1.295	3.018	3.064	3.163	3.211	3.164	3.212	3.164	3.212

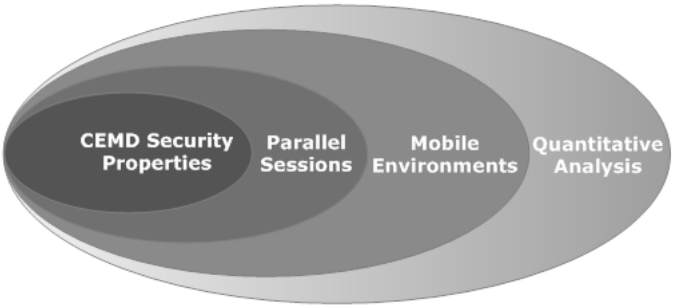
Time	20		40		60		80		100	
ber	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}	10^{-6}	10^{-3}
$n = 1$	1.931	1.939	2.019	2.027	2.020	2.028	2.020	2.028	2.020	2.028
$n = 2$	3.684	3.699	4.037	4.053	4.040	4.056	4.040	4.056	4.040	4.056
$n = 3$	5.162	5.182	6.050	6.074	6.060	6.084	6.060	6.084	6.060	6.084
$n = 4$	6.327	6.352	8.049	8.081	8.080	8.112	8.080	8.112	8.080	8.112
$n = 5$	7.206	7.234	10.021	10.060	10.100	10.140	10.100	10.140	10.100	10.140
$n = 6$	7.846	7.877	11.938	11.985	12.119	12.167	12.120	12.168	12.120	12.168
$n = 7$	8.288	8.321	13.767	13.821	14.136	14.193	14.140	14.197	14.140	14.197

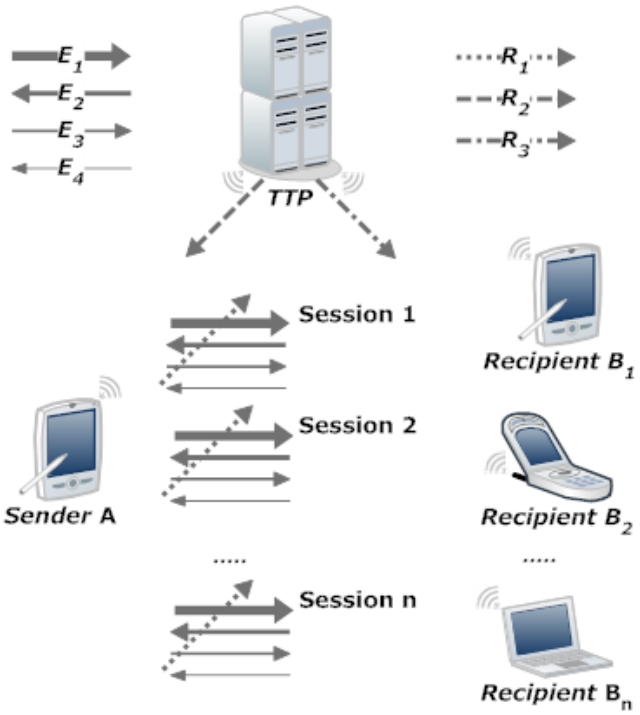
```
ctmc

module M1
v: [1..5] init 1;
[] (v=1) -> 1:(v'=2);
[] (v>1) & (v<5) -> 0.5:(v'=v-1) + 0.5:(v'=v+1);
[] (v=5) -> 1:(v'=4);
endmodule
```

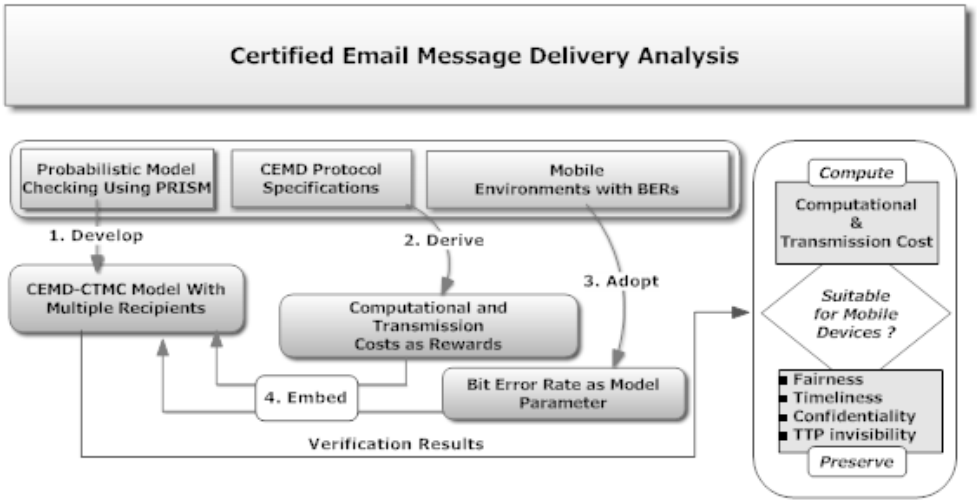
```
ctmc
// Number of recipients that A communicates with
const number_of_B_machines;
// BER parameter
const error_factor=1000;
const double error_env_wired=1;
const double ber=(error_env_wired/error_factor);
// Verifications
// Positive verification rates for messages Ei (where i = [1...4])
// Negative verification rate for Ei is defined as (1-Ei)
const double a;  // A's verification rate
const double b;  // Bs' verification rate
const double ver_E1=b;
const double ver_E2=a;
const double ver_E3=b;
const double ver_E4=a;
// Formula finish represents the final state of the model
formula finish = (TTP_certificates + no_TTP = number_of_B_machines);
...
```

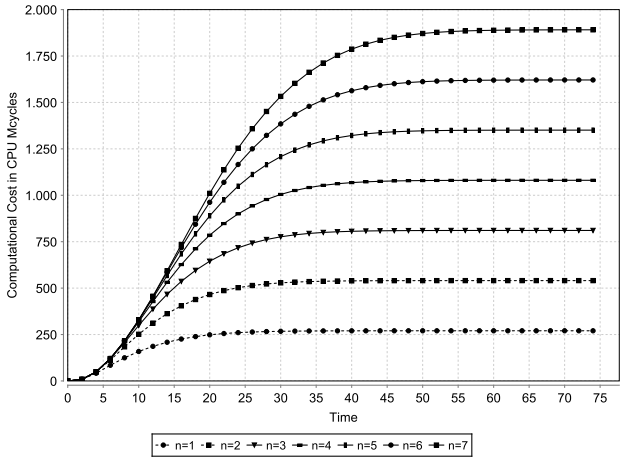
```
// Protocol's sender
module A
// ACKs sent by sender A to a recipient B_i
ACK_counter_A: [0..number_of_B_machines] init 0;
...
// A sends ACK to B_i
[A_send_ACK] (ACK_counter_A < number_of_B_machines) -> 1-ber: (
  ACK_counter_A'=ACK_counter_A+1) & (A_state'=1);
// Transmition error
[A_send_ACK_error] (ACK_counter_A < number_of_B_machines) -> ber: (
  partyA_state'=0);
...
end module
// Protocol's recipients
module B
// ACKs received by sender A to a recipient B_i
ACKS_received_B: [0..number_of_partyB_machines] init 0;
...
// B_i receives ACK from A
[A_send_ACK] (ACKS_received_B < number_of_partyB_machines) -> (
 ACKS_received_B'=ACKS_received_B+1) & (B_state'=1);
...
end module
```

```
    // Protocol's initiator
 2  module I
    ...
    // I sends E1
 5  [I_send_E1] (E1_queue_I < number_of_R_machines) & (E1_queue_I <
      ACKS_received_I) -> 1-ber: (I_state'=3) & (E1_queue_I'=E1_queue_I+1);
    // E1 Transmition error (BER)
 7  [I_send_E1_error] (E1_queue_I < number_of_R_machines) & (E1_queue_I <
      ACKS_received_I) -> ber: (I_state'=2);
    // E1 will be retransmitted after negative verification error
 9  [incorrect_ver_of_E1] (E1_queue_I > 0) -> (I_state'=2) &
      (E1_queue_I'=E1_queue_I-1);
    ...
11  endmodule

    // Protocol's responders
13  module R
    ...
    // R receives E1
16  [I_send_E1] (E1_queue_R < number_of_R_machines) ->
    (E1_queue_R'=E1_queue_R+1) & (R_state'=3);
    // Correct verification of E1
18  [verification_of_E1] E1_queue_R>0 & (E2_queue_R < number_of_R_machines)
    -> ver_E1: (R_state'=4) & (E2_queue_R'=E2_queue_R+1) &
    (E1_queue_R'=E1_queue_R-1);
    // Incorrect verification of E1
20  [incorrect_ver_of_E1] E1_queue_R>0 & (E2_queue_R < number_of_R_machines)
    -> 1-ver_E1: (R_state'=2) & (E1_queue_R'=E1_queue_R-1);
    ...
22  endmodule
```

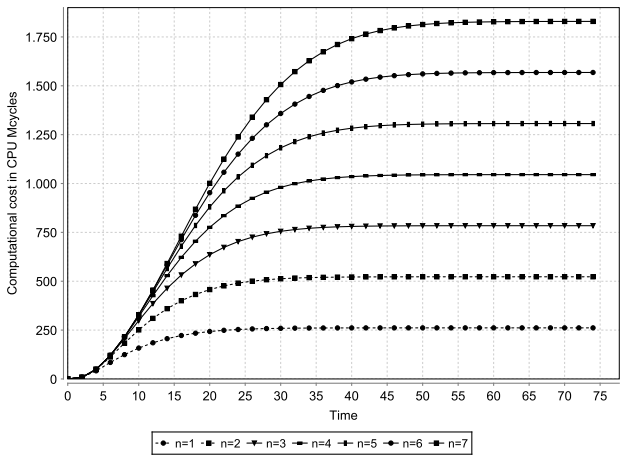

CEMD Security Properties
Parallel Sessions
Mobile Environments
Quantitative Analysis


E1
E2
E3
E4
R1'
R2'
R3'
TTP
Session 1
Session 2
Session n
Sender A
Recipient B1
Recipient B2
Recipient Bn

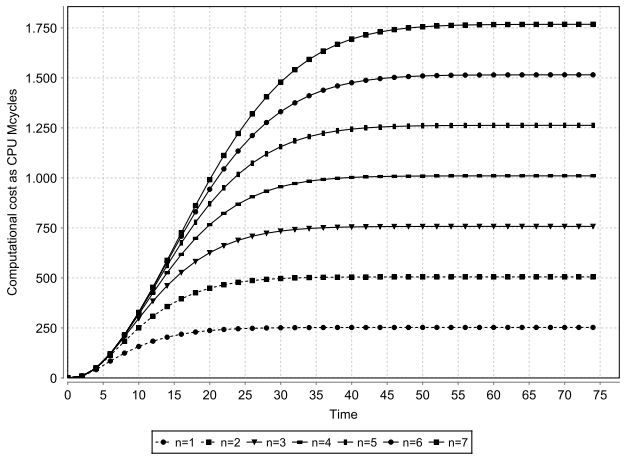

Certified Email Message Delivery Analysis
Probabilistic Model Checking Using PRISM
CEMD Protocol Specifications
Mobile Environments with BERs
1. Develop
2. Derive
3. Adopt
CEMD-CTMC Model With Multiple Recipients
Computational and Transmission Costs as Rewards
4. Embed
Bit Error Rate as Model Parameter
Verification Results
Compute
Computational & Transmission Cost
Suitable for Mobile Devices ?
Fairness
Timeliness
Confidentiality
TTP invisibility
Preserve


Computational Cost in CPU Mcycles
2.000
1.750
1.500
1.250
1.000
750
500
250
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

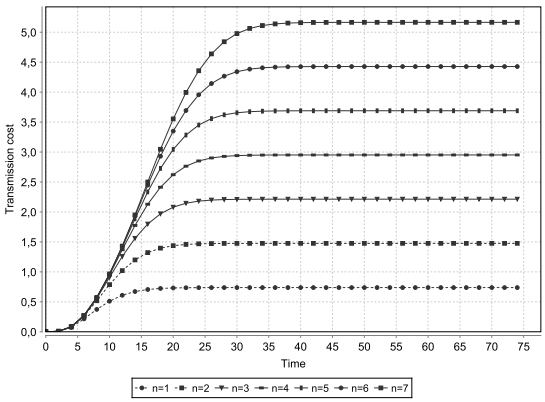
(a)


Computational cost in CPU Mcycles
1.750
1.500
1.250
1.000
750
500
250
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

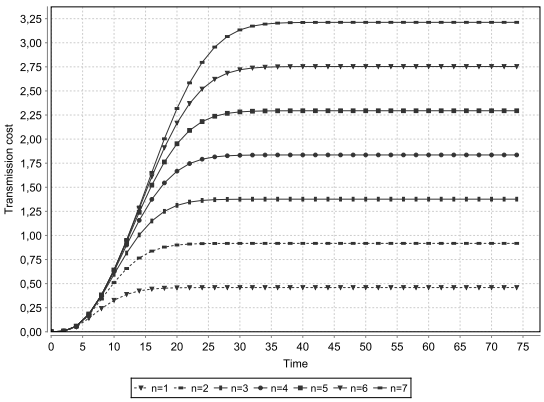
(b)


Computational cost as CPU Mcycles
1.750
1.500
1.250
1.000
750
500
250
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

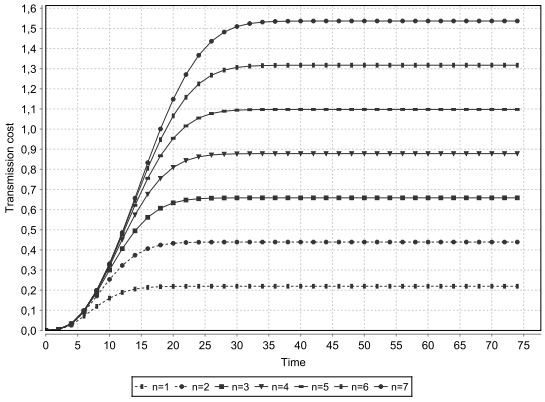
(c)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

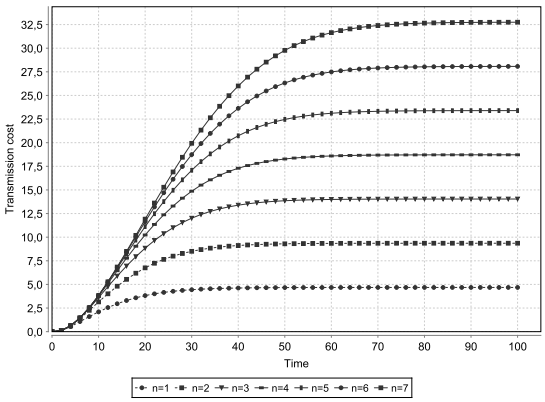
(a)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

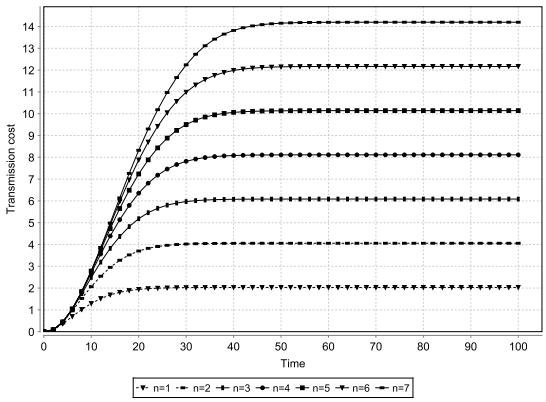
(b)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

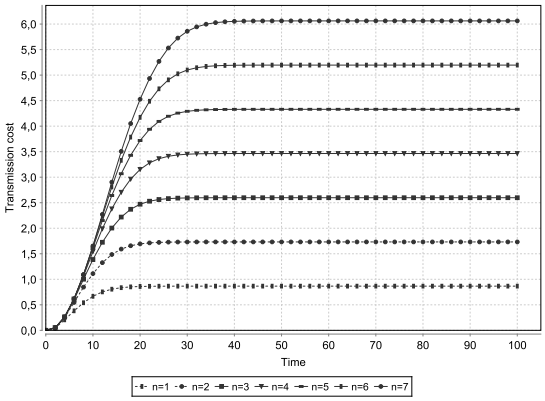
(c)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

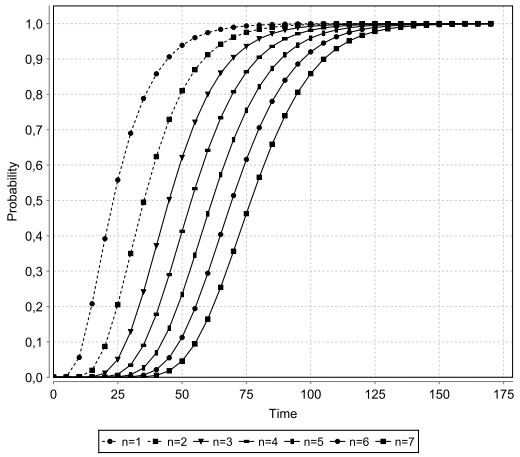
(a)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

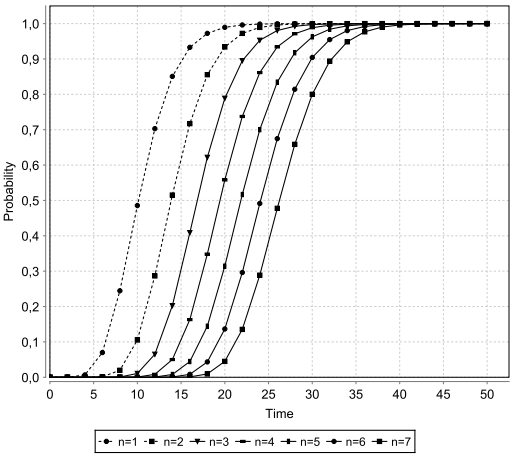
(b)


Transmission cost
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

(c)


Probability
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

(a)


Probability
Time
n=1 n=2 n=3 n=4 n=5 n=6 n=7

(b)