# AUTOMATED COMPARATIVE PERFORMANCE EVALUATION OF SECURITY PROTOCOLS

B. Genge, “Petru Maior” University of Tg. – Mureş, Romania

**Abstract -** We propose a comparative performance evaluation method for security protocols. We start by constructing a security protocol model where we assign a cost functions for each cryptographic operation. For each class of cryptographic operations (e.g. symmetric encryption, asymmetric decryption), we construct a polynomial function based on an exhaustive performance evaluation of cryptographic combinations including algorithms and key sizes. The proposed method is validated by a comparative analysis of 1000 generated protocols and 16 existing security protocols.

**Keywords** – Security protocols, performance evaluation, OpenSSL, Cryptlib, Crypto++.

**Abstract –** Propunem o metodă comparativă de evaluare a performanţelor protocoalelor de securitate. Începem cu construirea unui model al protocoalelor de securitate în cadrul căruia ataşăm funcţii cost pentru fiecare operaţie criptografică. Pentru fiecare asemenea clasă de operaţii (e.g. criptare simetrică, decriptare asimetrică), construim o funcţie polinomială prin evaluarea exhaustivă a tuturor combinaţiilor criptografice, printre care se numără algoritmi şi dimensiune a cheilor. Metoda propusă este validată prin analiza comparativă a 1000 de protocoale generate şi 16 protocoale existente.

**Cuvinte cheie –** Protocoale de securitate, evaluarea performanţelor, OpenSSL, Cryptlib, Crypto++.

## 1. Security protocols

Security protocols are “communication protocols dedicated to achieving security goals” (C.J.F. Cremers and S. Mauw) [1] such as confidentiality, integrity or availability. Achieving such security goals is made through the use of cryptography. Designing new protocols is a challenging task if we look at the number of attacks that have been discovered over the years [2] after the protocols have been published.

However, in the last few years the use of protocol composition [3, 4, 5] has been successfully applied to create new protocols based on existing [6, 7] or predefined protocols [3]. The composition process makes use of the informal [6] specification of security protocols which does not include any implementation-related information such as selected cryptographic algorithm, key size or encryption rounds. The result of the composition can have multiple protocols [8] from which the most performant must be selected. As mentioned earlier, cryptography is an important component of these protocols. This is why existing performance evaluation methods include several aspects related to the performance of the algorithms used to implement the protocols. However, in the composition phase, the cryptographic algorithms used in the implementation process are unknown. To help the decision process related to the selection of the most performant security protocol, in the early design phase, we propose a novel evaluation method that focuses on cryptographic algorithm operations, available in the informal specification. The informal specification does not include a formal tool for reasoning on security protocols. In order to achieve our goal, we need such a tool. We have chosen to use the strand space model [9] as a specification model because of its simplicity and extensibility. The rest of the paper is structured as follows. In Section 2 we present an extension of the original strand space used to model security protocols and we introduce the canonical model where cryptographic operations are modelled as t-strands having specific classifiers. In Section 3 we model cryptographic algorithms as polynomial functions. Using the proposed approach, we present several experimental results in section 4. In Section 5 we relate our work to others found in the literature. We end with a conclusion in Section 6.

## 2. K-strands and t-strands

A *strand* is a sequence of transmission and reception events used to model protocol participants. A collection of strands is called a *strand space*. The strand space model was introduced by Guttman et all in [9] and extended by the authors with participant knowledge, specialized basic sets and explicit *term* construction in [10]. The resulting model is called a *k-strand space*. The rest of this section formally defines the k-strand and k-strand space concepts. By analysing the protocol specifications from the SPORE library [11] we can conclude that protocol participants communicate by exchanging *terms* constructed from elements belonging to the following sets: R, denoting the set of participant names; N, denoting the set of nonces (i.e. “number once used”); K, denoting the set of cryptographic keys and M denoting user-defined components. If required, other sets can be easily added without affecting the other components. The above-defined basic sets and function names are used in the definition of *terms*, where we also introduce constructors for pairing and encryption:

$$\mathcal{T} ::= . \mid \mathsf{R} \mid \mathsf{N} \mid \mathsf{K} \mid \mathsf{M} \mid (\mathcal{T}, \mathcal{T}) \mid \{\mathcal{T}\}_{FuncName(\mathcal{T})},$$

where the ‘.’ symbol is used to denote an empty term. We use the symbol $\mathcal{T}^*$ to denote the set of all subsets of terms. To denote the transmission and reception of terms, we use *signed terms*. The occurrence of a term with a positive sign denotes transmission, while the occurrence of a term with a negative sign denotes reception. The set of transmission and reception sequences is denoted by $(\pm\mathcal{T})^*$.

**Definition 1.** *A* k-strand (i.e. knowledge strand) *is a tuple* $\langle \mathcal{K}, r, s \rangle$, *where* $\mathcal{K} \in \mathcal{T}^*$ *denotes the knowledge corresponding to the modelled participant,* $r \in \mathsf{R}$ *denotes the participant name and* $s \in (\pm\mathcal{T})^*$ *denotes the sequence of transmissions and receptions. A set of k-strands is called a k-strand space.*

As opposed to k-strands, in the t-strand model the terms exchanged between t-strands are based on *types* constructed from *basic typed terms* and are called *typed terms* or more simply *t-terms*:

$$\mathcal{T}_t ::= . \mid BasicTT \mid (\mathcal{T}_t, \mathcal{T}_t) \mid \{\mathcal{T}_t\}_{FuncName} .$$

**Definition 2.** *A* t-strand (i.e. typed strand) *is a tuple* $\langle c, r, s \rangle$, *with* $c \in C$, $r \in \mathsf{R}$ *and* $s \in (\pm\mathcal{T}_t)^*$. *A set of t-strands is called a t-strand space. The set of all t-strand spaces is denoted by* $\Sigma_t$.

### 3. Modelling cryptographic operations

Usually, protocol specifications do not include cryptographic operations such as term concatenation, encryption or signature generation, which are considered to be implementation-specific. However, when dealing with the performance evaluation of these protocols we can not omit such operations because they directly influence the evaluation process. By using the defined classifiers and typed strands, we can model cryptographic operations as follows.

**Definition 3.** *Let* $c \in C$ *be a classifier. Then the typed strands corresponding to this classifier generate the following sequences of transmissions and receptions for any* $t_t, t_t' \in \mathcal{T}_t$*:*

*Encryption.* $\langle -t_t, +\{t_t\}_{sk} \rangle$, *if* $c = C_{\mathcal{E}}$;

*Decryption.* $\langle -\{t_t\}_{sk}, +t_t \rangle$, *if* $c = C_{\mathcal{D}}$;

*Hash.* $\langle -t_t, +\{t_t\}_h \rangle$, *if* $c = C_{\mathcal{H}}$;

*Public key enc.* $\langle -t_t, +\{t_t\}_{pk} \rangle$, *if* $c = C_{\mathcal{PK}}$;

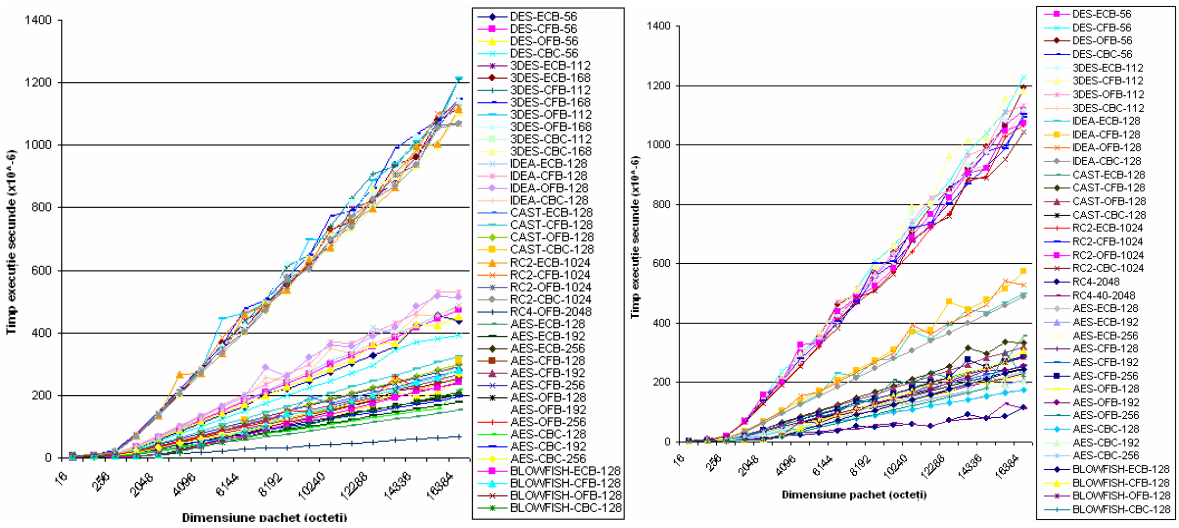
*Private key enc.* $\langle -t_t, +\{t_t\}_{pvk} \rangle$, *if* $c = C_{\mathcal{PVK}}$;

*Key-Gen.* $\langle +\mathrm{k} \rangle$, *if* $c = C_{\mathcal{K}}$;

*Nonce-Gen.* $\langle +\mathrm{n} \rangle$, *if* $c = C_{\mathcal{N}}$;

*Concatenation.* $\langle -t_t, -t_t', +(t_t, t_t') \rangle$, *if* $c = C_C$;

*Split.* $\langle -(t_t, t_t'), +t_t, +t_t' \rangle$, *if* $c = C_I$.




(a) (b)

Fig. 1. Execution time of cryptographic operations for: (a) Cryptlib (b) OpenSSL

Using the k-strand model, the operations that must be executed by protocol participants are extracted and the t-strand model is constructed. The extraction process uses the knowledge corresponding to each k-strand to identify operations. Thus, terms that are not in the participant's knowledge must be generated (i.e. keys, random numbers) or extracted from encrypted terms which are also located in the knowledge set. In the t-strand model, the t-nodes responsible for creating new t-terms have a positive sign. Thus, we assign a cost to each positive t-node found in a t-strand space. The functions corresponding to each cryptographic operation type have been constructed using an exhaustive performance analysis of cryptographic algorithms from two well-known cryptographic libraries: Cryptlib [13] and OpenSSL [14]. For example, the performance of symmetric algorithms corresponding to the two cryptographic libraries is given in figure 1. From our experimental results, we reached the conclusion that cryptographic algorithm classes can be approximated using the polynomial function:

$$f(x) = \alpha_4 x^3 + \alpha_3 x^2 + \alpha_2 x + \alpha_1 . \tag{1}$$

For each algorithm class we need to solve the set of equations:

$$\begin{cases} \dfrac{\partial R}{\partial \alpha_1} = -2\sum_{i=1}^{n}\left(y_i - \left(\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1\right)\right) = 0 \\ \dfrac{\partial R}{\partial \alpha_2} = -2x_i\sum_{i=1}^{n}\left(y_i - \left(\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1\right)\right) = 0 \\ \dfrac{\partial R}{\partial \alpha_3} = -2x_i^2\sum_{i=1}^{n}\left(y_i - \left(\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1\right)\right) = 0 \\ \dfrac{\partial R}{\partial \alpha_4} = -2x_i^3\sum_{i=1}^{n}\left(y_i - \left(\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1\right)\right) = 0 \end{cases} \tag{2}$$

Part of the graphical representation of these polynomials is given in figure 2.

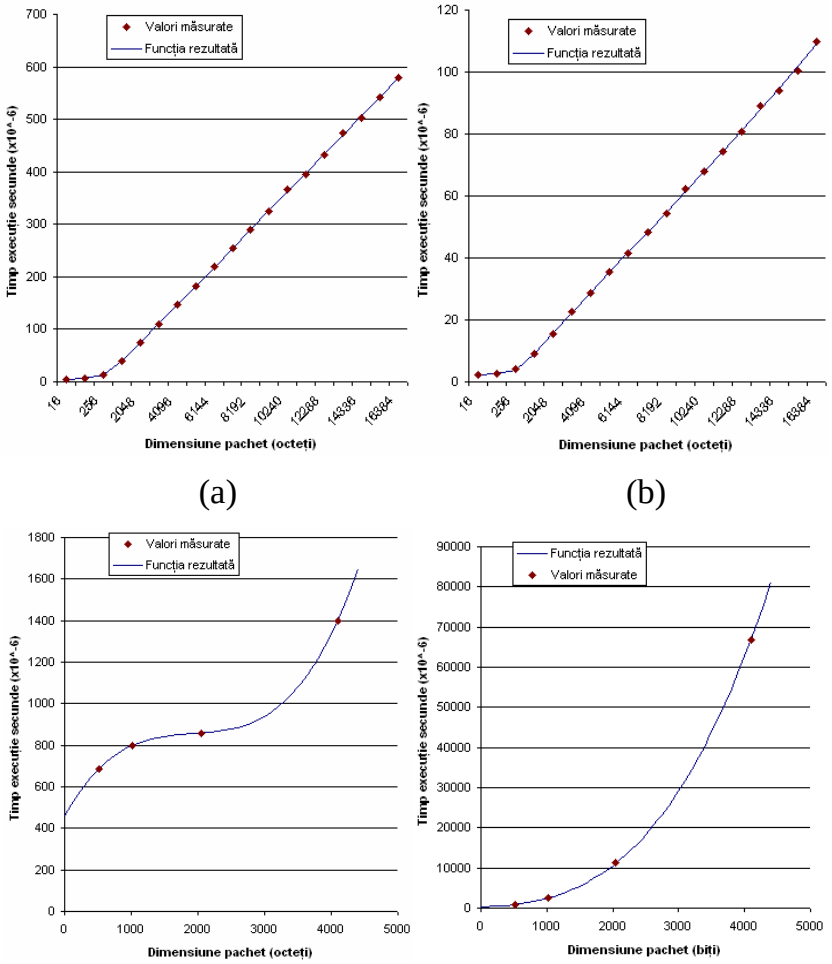



Fig. 2. Graphical representation of the algorithm models: (a) Symmetric encryption (b) Hash (c) Asymmetric encryption (d) Signature

## 4. Experimental results

Based on the calculated models and cryptographic operations, we have generated 1000 security protocol specifications. We have compared the performance of these protocols to the estimated performance using protocol pairs. As we can see from figure 3, the estimation strictly follows the measured performance. In some case, the predicted values do not correspond to the measured ones. These situations are marked with black arrows. This is because the measured protocol performances are very similar (under 1 milliseconds), which, in reality, does not affect the performance of the implementing system. As we can also see from figure 4, the estimation error depends on the package size used to calculate the estimated performance. Because of this, the error decreases in value as package size grows. We have also provided a comparative performance evaluation of real security protocols. The experimental results are given in table 2. As illustrated in table 1, the estimated performances of similar protocols does not correspond to the measured values, which is illustrated using emphasized text.

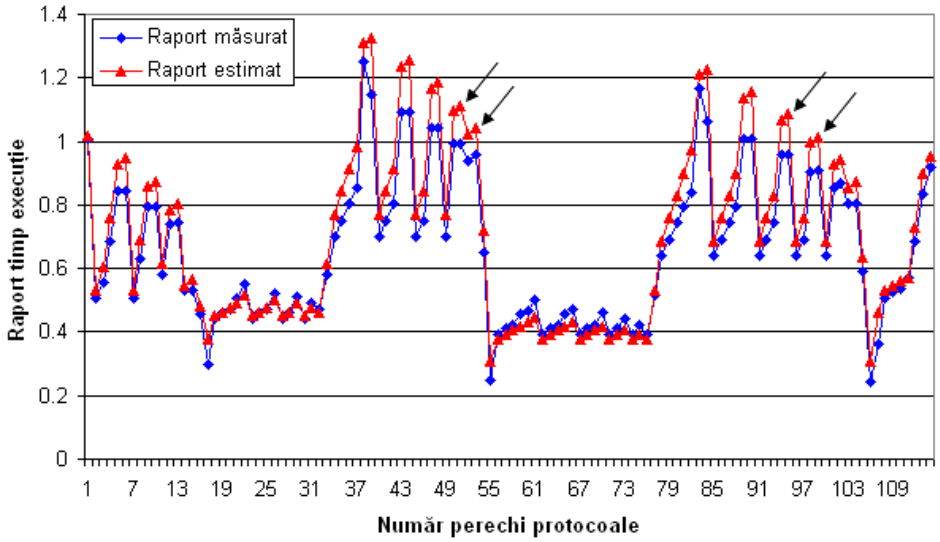



Fig. 3. Graphical representation of the predicted and measured protocol performance pairs

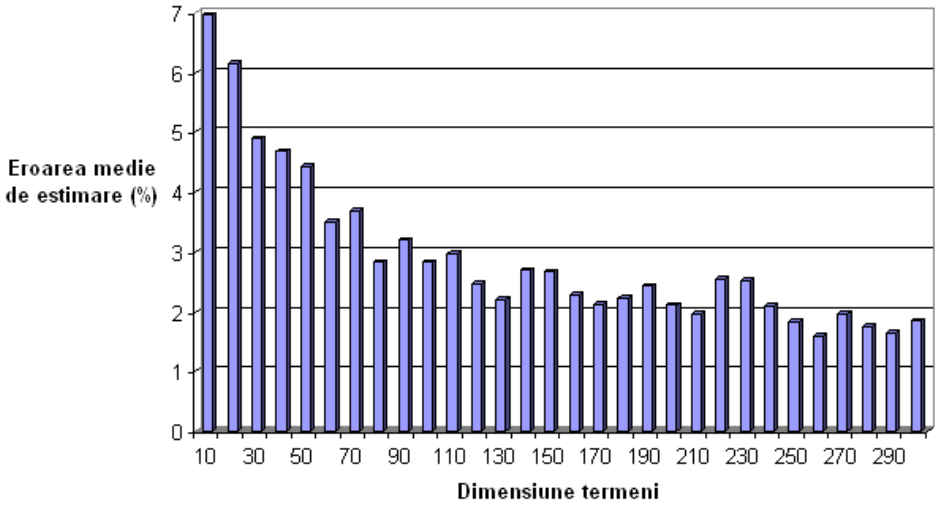



Fig. 4. Average estimation error based on the estimation package size

**Table 1.** Comparative performace evaluation of protocols from SPORE

| | SEC-RPC | BAN-RPC | BANC-RPC | LOWE-RPC | CCITTv1 | CCITTv1c | CCITTv3 | BAN-CCITT | DENNING-SACCO | LOWE-DENNING | KAO-CHOWv1 | KAO-CHOWv2 | KERBEROS v5 | NEEDH-SCH-PK | YAHALOM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S-RPC | - | **0.94** | 1.78 | 1.78 | 0.02 | 0.01 | 0.01 | 0.01 | 0.82 | 0.6 | 0.6 | 0.51 | 0.37 | 0.01 | **0.8** |
| B-RPC | **1.06** | - | 1.89 | 1.89 | 0.02 | 0.01 | 0.01 | 0.01 | 0.87 | 0.64 | 0.64 | 0.54 | 0.39 | 0.01 | 0.85 |
| C-RPC | 0.56 | 0.53 | - | 1.0 | 0.01 | 0.0 | 0.01 | 0.01 | 0.46 | 0.34 | 0.34 | 0.29 | 0.21 | 0.01 | 0.45 |
| L-RPC | 0.56 | 0.53 | 1.0 | - | 0.01 | 0.0 | 0.01 | 0.01 | 0.46 | 0.34 | 0.34 | 0.29 | 0.21 | 0.01 | 0.45 |
| CCv1 | 46.2 | 43.5 | 82.2 | 82.2 | - | 0.33 | **0.41** | **0.42** | 38.0 | 27.9 | 27.9 | 23.5 | 17.0 | 0.49 | 37.0 |
| CCv1c | 139 | 131 | 247 | 247 | 3.01 | - | 1.25 | 1.28 | 114 | 84.1 | 84.2 | 70.8 | 51.3 | **1.48** | 111 |
| CCv3 | 111 | 105 | 198 | 198 | **2.41** | 0.8 | - | 1.02 | 91.7 | 67.4 | 67.4 | 56.7 | 41.1 | 1.19 | 89.3 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B-CC | 108 | 102 | 193 | 193 | **2.35** | 0.78 | 0.98 | - | 89.6 | 65.8 | 65.8 | 55.4 | 40.1 | 1.16 | 87.2 |
| D-S | 1.22 | 1.14 | 2.16 | 2.16 | 0.03 | 0.01 | 0.01 | 0.01 | - | 0.74 | 0.74 | 0.62 | 0.45 | 0.01 | **0.97** |
| L-D-S | 1.65 | 1.56 | 2.94 | 2.94 | 0.04 | 0.01 | 0.01 | 0.02 | 1.36 | - | **1.0** | 0.84 | 0.61 | 0.02 | 1.32 |
| K-Cv1 | 1.65 | 1.56 | 2.94 | 2.94 | 0.04 | 0.01 | 0.01 | 0.02 | 1.36 | **1.0** | - | 0.84 | 0.61 | 0.02 | 1.32 |
| K-Cv2 | 1.96 | 1.85 | 3.49 | 3.49 | 0.04 | 0.01 | 0.02 | 0.02 | 1.62 | 1.19 | 1.19 | - | 0.72 | 0.02 | 1.57 |
| KERv5 | 2.71 | 2.55 | 4.82 | 4.82 | 0.06 | 0.02 | 0.02 | 0.02 | 2.23 | 1.64 | 1.64 | 1.38 | - | 0.03 | 2.17 |
| NS-PK | 93.7 | 88.2 | 166 | 166 | 2.03 | **0.67** | 0.84 | 0.86 | 77.1 | 56.6 | 56.6 | 47.7 | 34.5 | - | 75.0 |
| YAH | **1.24** | 1.18 | 2.22 | 2.22 | 0.03 | 0.01 | 0.01 | 0.01 | **1.02** | 0.75 | 0.75 | 0.64 | 0.46 | 0.01 | - |

## 5. Related work

In the literature we find several papers dealing with the performance evaluation of protocol implementations [15, 16]. In contrast, only a few papers are dedicated to constructing a model for the evaluation of security protocol performance [6, 12]. For completeness, we first mention a few papers that adopted the performance evaluation of various cryptographic algorithm and security protocol implementations. In [15] and [16], the performance of cryptographic algorithms is measured as a function of the total amount of energy consumed by the device on which the algorithm is running. For evaluating the performance of the WTLS [18] (Wireless Transport Layer Security) protocol, the authors from [19] measure the time needed to perform connections on a PDA. Finally, we mention the currently world wide adopted security protocol, TLS [17] (Transport Layer Security). The performance of TLS has been intensively studied [20, 21]. The results show that the cryptographic overhead introduced by TLS seriously affects the performance of regular servers. Because of this, several solutions have been proposed to improve server performance, from which we mention the distribution of cryptographic operations among other servers [21] and the use of hardware accelerators [22]. One of the papers dedicated to modelling the behaviour of protocol components [12] constructs a parametric mathematical model based on an exhaustive evaluation process of algorithm implementations. The constructed model does not address, however, the issue of protocol cryptographic operations executed by participants. A similar approach to ours is proposed in [6] where cryptographic operations are detailed and each operation is assigned a symbolic cost. Our approach differs by the fact that it introduces the concept of size based on term types, as opposed to instance values used in [6]. In addition, we also model the size of message components resulting from cryptographic operations, which is not covered in [6].

## 6. Conclusion and future work

We have developed a procedure for evaluating the performance of security protocols. Our proposal is based on a canonical model which eliminates terms specific to protocol instantiations, leaving only types. The canonical model also includes cryptographic operations that must be executed by protocol participants in order to construct new terms. The total cost associated to cryptographic operations denotes the performance of the analysed security protocol. The novelty of our approach lies in the use of participant knowledge to construct cryptographic operations, which does not need any

user intervention and provides a minimal effort from participants to create protocol messages. Another novelty introduced by our approach is the association of typed terms to symbolic sizes and the modelling of ciphertext size resulting from cryptographic operations. As future work, we intend to introduce additional cryptographic operations denoting the verification of received terms. We also intend to use the proposed performance evaluation method in the composition process, which has been used as a method for designing new security protocols from existing protocols. Thus, designers could chose from an early stage the most performant protocol.

## References


[1] C. Cremers, S. Mauw - *Checking secrecy by means of partial order reduction*, In S. Leue and T. Systa, editors, Germany, september 7-12, 2003, revised selected papers LNCS, Vol. 3466, 2005, Springer.

[2] Gavin Lowe - *Some new attacks upon security protocols*, In Proceedings of the 9th Computer Security Foundations Workshop, IEEE Computer Society Press, 1996, pp. 162-169.

[3] Hyun-Jin Choi - *Security protocol design by composition*, Cambridge University, UK, Technical report Nr. 657, UCAM-CL-TR-657, ISSN 1476-2986, 2006.

[4] Ran Canetti - *Universally composable security: A new paradigm for cryptographic protocols*, 42nd FOCS, 2001, Revised version (2005), available at eprint.iacr.org/2000/067.

[5] Cas J. F. Cremers - *Compositionality of Security Protocols: A Research Agenda*, Electr. Notes Theor. Comput. Sci., 142, pp. 99-110, 2006.

[6] A. Datta, A. Derek, J. C. Mitchell, A. Roy - *Protocol Composition Logic (PCL)*, Electronic Notes in Theoretical Computer Science Volume 172, 1 April, 2007, pp. 311-358.

[7] S. Andova, Cas J.F. Cremers, K. Gjosteen, S. Mauw, S. Mjolsnes, and S. Radomirovic - *A framework for compositional verification of security protocols*, Elsevier, to appear, 2007.

[8] B. Genge, P. Haller, R. Ovidiu, I. Ignat - *Term-based composition of security protocols*, In the Proceeedings of the 16th International Conference on Automation, Quality and Testing, Robotics, AQTR, 2008, pp. 233-238.

[9] F. Javier Thayer Fabrega, Jonathan C. Herzog, Joshua D. Guttman - *Strand spaces: Proving security protocols correct*, Journal of Computer Security 7, 1999, pp. 191-230.

[10] Genge Bela, Iosif Ignat - *Verifying the Independence of Security Protocols*, IEEE 3rd International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2007, pp.155-163.

[11] *SPORE, Security Protocol Open Repository*, http://www.lsv.ens-cachan.fr/spore.

[12] Phongsak Kiratiwintakorn - *Energy efficient security framework for wireless Local Area Networks*, PhD Thesis, University of Pittsburgh, 2005.

[13] *Cryptlib Software Distribution*, http://www.cs.auckland.ac.nz/~pgut001/cryptlib/ [last access April, 2008].

[14] *OpenSSL Software Distribution*, http://www.openssl.org/ [last access April 2008].

[15] M. Viredaz and D. Wallach - *Power evaluation of a handheld computer: A case study*, Compaq Western Research Lab, Tech. Rep. 2001/1, 2001.

[16] S. Hirani - *Energy efficiency of encryption schemes in wireless devices*, Master's thesis, Telecommunications Program, University of Pittsburgh, Pittsburgh, 2003.

[17] Dierks, T,. Allen, C. - *The TLS Protocol, Version 1.0, Request for Comments: 2246*, Network Working Group, January 1999.

[18] WAP Forum - *Wireless Transport Layer Security Specification Version 1.1, 11.2.*, 1999.

[19] Neil Daswani - *Cryptographic Execution Time for WTLS Handshakes on Palm OS Devices*, Certicom Public Key Solutions, September 2000.

[20] Cristian Coarfa, Peter Druschel and Dan S. Wallach - *Performance Analysis of TLS Web Servers*, ACM Transactions on Computer Systems, 24 (1), 2006, pp. 39-69.

[21] Adam Stubblefield, Aviel D. Rubin, Dan S. Wallach - *Managing the Performance Impact of Web Security*, Electronic Commerce Research, No. 5, Springer Science + Business Media, 2005, pp. 99–116.

[22] D. Dean, T. Berson, M. Franklin, D. Smetters, and M. Spreitzer - *Cryptology as a network service*, In Proceedings of the 7th Network and Distributed System Security Symposium, San Diego, California, Feb. 2001.