Decision Process for Automated Selection of Security Protocols

Lars Völker, Christoph Werle, Martina Zitterbart
Institut für Telematik
Universität Karlsruhe (TH), Germany

***Abstract*— Today's Internet has a growing number of protocols and mechanisms to protect data in transmission. One can choose from IP Security (IPsec), Transport Layer Security (TLS), and many other protocols. However, available security protocols and mechanisms are not widely used due to usability issues [1], [2] and because users often underestimate the risk their data is exposed to. An approach to solve this problem consists of automated selection and configuration of available security protocols in a user-transparent way. In this paper, we present a method for automatically choosing the right security protocol based on Security, Quality of Service, and Energy Consumption aspects. We describe the necessary aspects, value functions, and a hierarchical, flexible, and efficient decision process.**

I. Introduction

With the present multitude of security protocols and countless configuration and misconfiguration options, users are often challenged to take decisions they do not fully understand. In order to avoid this situation, our vision is to automate the process of choosing and configuring a security protocol appropriate to the needs of the user. To enable this automated approach, a set of policies needs to be given and information on a user's preferences is required.

However, one cannot just add a security protocol to a network connection without the security protocol influencing others aspects as Quality of Service and Energy Consumption. In previous work [3], we have shown that Security and Quality of Service influence each other and cannot be taken into account separately. This is also true for the selection of security protocols, which is the subject of this paper. In addition, we also consider energy consumption, since the cryptographic algorithms used to provide security add a significant computational overhead, which in turn results in increased energy consumption.

In this paper, we focus on the security protocol selection process, which basically consists of the following steps:

- Collecting all available candidates.
- Filtering out candidates based on requirements.
- Evaluating each candidate.
- Selecting candidate with highest rating.

Candidates are instances of security protocols, for example cipher suites of TLS/SSL[1]. For a given connection, the system collects all applicable candidates by checking all available security protocols for support of protecting the connection under consideration. We discuss the detection of applicable candidates detailed in Section II. The filtering of candidates is based on requirements specified by administrative policies, e.g., minimal acceptable key length or required bandwidth. In the next step each candidate is rated. Our solution is based on the multi-attribute utility theory (MAUT) and follows a hierarchical approach as introduced in Section III. At first, for each category (Security, Quality of Service, Energy Consumption) ratings are calculated, which are subsequently aggregated to an overall rating. After having filtered and ranked all candidates, one just needs to choose the candidate with the highest rating, which is the security protocol with the highest utility to the user. These steps are presented in detail in Section III.

One difficulty in constructing such a process is finding comparable parameters for each category and then rating them. In Section IV we present our criteria and in Section V we show some exemplary value functions to rate these criteria. Furthermore, the process must allow for aggregation of ratings of different categories, e.g., Security and Quality of Service, in order to find a meaningful overall rating. This is presented in Section VI. We close this paper with Conclusion and Future Work in Section VIII.

In this paper, we present our solution for the automated selection of security protocols. The contributions of this paper include the automated, efficient, hierarchical, and flexible process for choosing a security protocol for a given connection, based on policy and requirements. We identify important Security, Quality of Service, and energy consumption properties, and define functions to map the respective attribute values to utility ratings. We keep our solution modular, allowing for the inclusion of additional properties, and flexible by allowing for adjustment of the process using weights and parametrized functions.

II. The System

This paper focuses on the selection of security protocols based on criteria defined in this paper. Since we implemented this process for selection of security protocols as an integral part of the Auto-Configuration of Communication Security (ACCS) system, this section will give a brief overview of the system.

The major goal of ACCS is to assist users by automatically selecting and configuring security protocols based on (administratively) defined policies. Once configured, ACCS is able

[1]The cipher suite describes the security protocol configuration as a possible choice in the protocol negotiation and states, in the case of TLS/SSL, the algorithm for key exchange, authentication, data encryption, and data authentication. In addition the key sizes of the algorithms are included.

to decide, whether a connection needs to be protected. If so, ACCS autonomously detects, which locally available security protocols, for example IPsec, TLS, or any other standardized security protocols, are supported by the communication partner. If none of the detected alternatives do comply with the policy applicable to the connection, ACCS prevents accidental data leakage by keeping the communication from taking place. If there are policy-conforming alternatives available to secure the connection, the alternative which most of all suits the preferences of the user, is selected and—transparently to the user—applied to the connection.

Figure 1 presents the basic architecture of ACCS. The central component is the Security Manager, which basically is the decision making component. Using the Detection Modules, the Security Manager can detect important events, e.g., new network connections, and decide, whether action has to be taken. This decision is based on policies stored in the *Policy* component, which can be accessed by applications (allowing for per-connection policy specification) and administrators (allowing for modification of global policies, which overrule any user-given policy). The *Evaluation Modules* represent decision algorithms and the *Knowledge Base* stores needed information for the selection process.

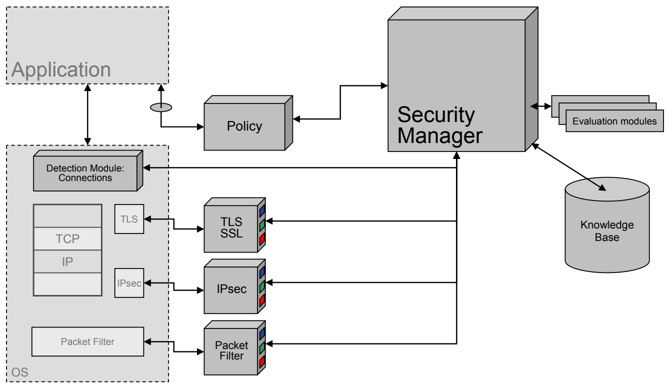

Fig. 1. ACCS Architecture

The Security Manager is accompanied by adapters used to connect to a available Security Protocols. In Figure 1 *TLS/SSL*, *IPsec*, and *Packet Filter* are depicted as exemplary adapters. While the Packet Filter is not a Security Protocol in the traditional sense, it can be used to influence the observed traffic, e.g., by dropping packets.

The basic operation of ACCS starts out with an event of a detection module, e.g., a new socket is opened and therefore a new network connection is about to be established. This event is received by the Security Manager and, based on the specified policy, the Security Manager decides whether and how to take action. If action has to be taken, the Security Manager contacts the Adapters to gather information about locally available Security Protocols, which are supported by the remote system, and how they can protect the network connection using the description carried by the detection event. Each Adapter can return zero or more protection options, called candidates for the remainder of this paper. Commonly, a security protocol does not return only one candidate but a list of candidates based on different options it has. The TLS/SSL adapter, for example, will usually return the list of available cipher suites. The selection process works on these returned candidates and instructs ACCS which action to take. The adapters of ACCS will finally implement the required changes to the communication traffic.

III. Our Approach

In this section, we give an overview of our approach before detailing individual parts in the following sections.

In order to select the "best" solution, the aforementioned steps have to be taken. *Collecting of candidates* is performed by ACCS, as presented in Section II. To *filter out candidates* and to *evaluate each candidate*, one has to describe the candidates in a suitable way, i.e., enabling comparability between candidates. We focus on the following aspects: Security, Quality of Service, and Energy Consumption. Section IV goes into details of the description of the criteria used to differentiate between candidates.

Additionally, a description of the term "best" is needed, which may vary depending on many aspects, e.g., the user, the user's location, or the service being used. The problem of selecting the "best" alternative out of a set of possible alternatives, based on defining and differentiating attributes, i.e., criteria, is well known in decision theory. Those multi-attribute problems can be dealt with using a variety of methods, for example the Analytic Hierarchy Process [4], ideal point methods as TOPSIS [5] or ELECTRE [6], or *multi-attribute utility theory* (MAUT) [7]. Our approach uses the afore-mentioned *MAUT*, which will be sketched in this section, to calculate a rating for each available candidate.

However, being able to describe the candidates (a_i) using criteria (c_j) and choosing a decision-making method is not enough; one also needs to define how to find the "best" solution in a given situation. This information is stored in the Policy component introduced in Section II. A policy can define *requirements* to omit unsuitable candidates, and values to parametrize the decision process: *weights* and *parameters*. The *weights* (w_j) are used to control the relative importance of the criteria to the overall rating. The *parameters* are used to adjust the *value functions*. For example: the parameters used in value function v_j influence the resulting utility of criterion c_j. Using MAUT, for each criterion c_j, $1 \leq j \leq m$, a (potentially parametrizable and therefore reusable) value function v_j needs to be defined, which maps the attribute value of alternative a_i with regard to criterion c_j, i.e. $c_j(a_i)$, to a utility value representing the actual usefulness of the attribute value to the user. Some specific value functions for the relevant criteria (Security, Quality of Service, Energy Consumption) are presented in Section V.

To generate an overall rating, we use the weighted sum of the single utility values to aggregate the individual utility values, which yields the following equation:

$$v(a_i) = \sum_{j=1}^{m} w_j * v_j\big(c_j(a_i)\big) \qquad (1)$$

This gives us a weighted sum over all criteria utilities using dedicated value functions for each criterion. We decided to use the weighted sum for aggregation as it is intuitive, and more complicated aggregation methods are not required in this context. Having calculated this *total utility value* of each candidate, we are now able to compare candidates.

Taking into account the problem and design space, we opt to use a hierarchical approach. We calculate the weighted sum of each aspect (Security, Quality of Service, Energy Consumption) and calculate the overall weighted sum afterwards. This hierarchy is further detailed in Section IV.

The notions and abbreviations are summarized in Table I.

Meaning	Abbreviation
Alternative i	a_i
Criterion j	c_j
Weight for criterion j	w_j
Attribute value of a_i with regard to criterion c_j	$c_j(a_i)$
Value function for criterion c_j	v_j
Total value of alternative a_i	$v(a_i)$

TABLE I
ABBREVIATIONS

IV. Criteria

In this section, we present our description of security mechanisms, which forms the foundation for the subsequent rating. The selection of a certain security mechanism does not only influence the security of the communication, but also leads to multiple implications in terms of other aspects, e.g., Quality of Service and Energy Consumption. The main reason for this influence is the computationally expensive cryptography. The calculation of cryptographic algorithms uses additional memory and CPU cycles, and therefore electrical power. It also introduces latency and latency jitter into packet processing and possibly limits achievable throughput. For this reason, we choose to include Quality of Service and Energy Consumption in the following description of criteria. All of the following criteria have to be reported by the security adapter, which detected the corresponding candidate. The measurement of some of the criteria, e.g., energy consumption, is very challenging and presents an active research topic. As the measurement of these values is not a central point of this paper, we will only briefly sketch available methods later on.

In the following sections, criteria used to describe alternatives are presented with a short motivation of their use. We organize the criteria hierarchically as depicted in Figure 2, which also gives a rough overview over the criteria used.

A. Security

Measuring and comparing security or defining a degree of security is a difficult task. Cryptographic primitives, e.g.,

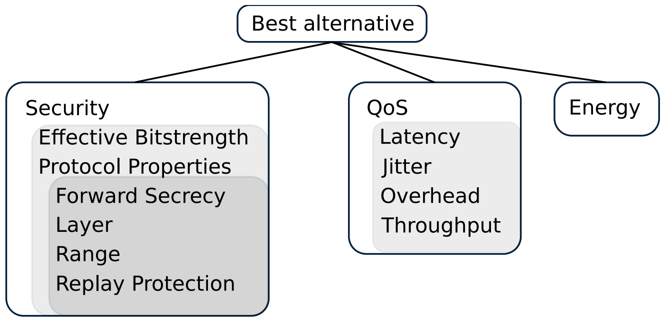

Fig. 2. Hierarchy of criteria

encryption algorithms, provide a certain amount of security depending on the algorithm used and important configuration parameters such as the key length in use. Although the key length is an often used metric to compare the security of cryptographic primitives, it does not reflect the security provided by a primitive well. The key length allows for estimation of resistance against brute force attacks, but attackers must be supposed to use the currently best known attack on any primitive under consideration. As the assessment of the quality of a cryptographic algorithm is a complex and error-prone task, we will resort to using existing recommendations. Many governmental and scientific institutions regularly publish recommendations on the strength of cryptographic primitives. Therefore, to compare cryptographic primitives, we will use the effective bitstrength as presented in the next section. Afterwards we consider protocol-specific security aspects.

1) Effective Bitstrength: The effective bitstrength or comparable algorithm strength, as defined by the NIST Computer Security Division [8], defines a comparison measure for cryptographic algorithms taking the best currently known attack into account. AES for example is currently supposed to provide an effective bitstrength equivalent to the actual key length used, whereas the Triple DES algorithm, using three independent keys with a key length of 56 bit, is supposed to provide an effective bitstrength of about 112 bit. Analogous estimations of the strength provided by cryptographic algorithms are also given for primitives based on e.g., Integer Factorization Cryptography (IFC), Finite Field Cryptography (FFC), and Elliptic Curve Cryptography (ECC).

The cryptographic primitives we take into consideration when evaluating the overall effective bitstrength of a security protocol are:

- Key Exchange
- Authentication
- Encryption
- Message Authentication

Depending on the security demand of the user and the behavior of an expected attacker—as defined by the attacker model—not all of the primitives listed above are necessary to provide adequate protection. If a user only wants to assure the integrity of exchanged messages, the strength of an encryption algorithm, if present at all, will be rather irrelevant to the user.

As a further example, when assuming a passive attacker, only key exchange and message authentication are potential targets and, therefore, are the only primitives included in an assessment of the security protocol.

For the aforementioned reasons, we give users the possibility to define, which of the four primitives defined above they actually need and consequently want to be taken into account. To determine the effective bitstrength ebs_i of a security mechanism a_i under inspection, we use the minimum operator on the individual effective bitstrength of the primitives demanded by the user[2]:

$$ebs_i = min(ebs_{i,keyex}, ebs_{i,auth}, ebs_{i,enc}, ebs_{i,mac}) \quad (2)$$

This estimation of the effective bitstrength of a security mechanism ebs_i forms the first criterion for the later utility analysis.

2) Protocol Properties: In this section we identify generic parameters to describe important, security-relevant parameters of network protocols. We will use the following criteria for description of protocol properties:

Layer
: The ISO-/OSI layer at which a security protocol is located determines, which protocols and which part of the protocol data unit the cryptographic mechanisms are applied to.

Range
: The range describes how far a security protocol reaches. It could only protect as far as one hop as in WEP or provided an end to end protection as in TLS. Figure 3 provides two examples: The communicating applications on Node A and Node B exchange data. Applying a security protocol working on the data link layer, e.g., WEP, offers protection for layers above the data link layer but only over one hop. IPsec in transport mode provides protection for layers above the network layer and in this example end-to-end protection.[3]

Forward Secrecy
: To differentiate between varying degrees of forward secrecy, this property is assigned to the protocol. When using the Diffie-Hellman (DH) key exchange, for example, the protocol determines when fresh exponents are to be generated and therefore determines the degree of forward secrecy provided. We differentiate between the following degrees:

none Both communication partners always use the same DH exponents.

partial One communication partner uses a fresh DH exponent whereas the other one uses a static one.

full Both communication partners generate a fresh DH exponent.

TLS, for example, allows DH with static/long-lasting keys and DH with ephemeral/short-time keys. In the first case TLS can only guarantee partial Forward Secrecy, since the DH exponents (of at least one communication partner) are used for a very long time and stored on the server. Other protocols, like IPsec, might even dynamically reuse DH exponents when encountering high load. This, however, might not negatively affect the Forward Secrecy.

Replay Protection
: Boolean parameter describing whether the protocol provides replay protection—replay protection keeps the attacker from successfully replaying authenticated messages into the communication.

[2]For the context of this paper, the equation is slightly simplified and does not show how to completely leave security primitives out of the evaluation. However, it is possible to either set the given effective bitstrength $ebs_{i,x}$ of the primitive x to 1 or introduce boolean factors for each primitive.

[3]Please note that security protocols can commonly only protect protocols on top of them.

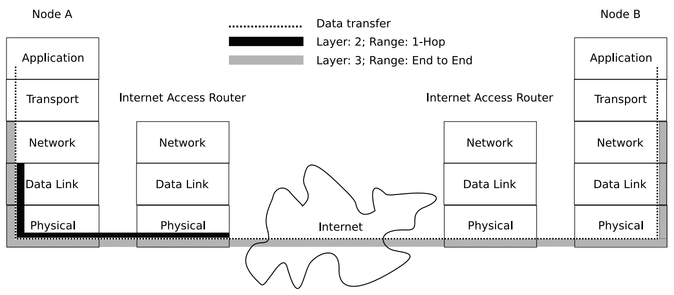

Fig. 3. Protocol properties: Range and Layer

B. Quality of Service

As shown at the beginning of this section, the choice of a security mechanism does affect the Quality of Service parameters of a connection. To model this behavior we will use the following criteria:

Latency
: Applying cryptographic operations introduces a certain amount of latency in the packet processing. This latency, e.g., arises from necessary computations, memory allocations, and bus transfers.

Latency jitter
: Additionally, the introduced latency is subject to jitter, especially in non real-time operating systems. In the common case, there are many processes competing for resources with hard predictable scheduling properties. This introduces hard to predict latency jitter.

Throughput
: The throughput achievable with a security protocol is limited by the link capacity or alternatively the processing speed of the cryptographic mechanisms to be applied. The achievable throughput is therefore used as a criterion.

Overhead
: The overhead caused by additional packet headers added by the security protocol in number of bytes.

Latency, latency jitter, and throughput also depend on the size of the data to which cryptographic operations have to be applied. As a simple example consider only two alternatives to perform cryptographic calculations: the standard CPU and a cryptographic accelerator card. As shown in comparisons [9], for small packet sizes the standard CPU often delivers better performance than dedicated cryptographic hardware. This effect is due to the high overhead of necessary bus accesses to transfer data to and from the cryptographic hardware. To model this behavior, we use a traffic profile concept and allow the user to assign this traffic profile to a policy. The traffic profile consists of three classes: small, medium, and large packets, as defined in Table II. Given the estimated probabilities p_x for each packet size x of an application's packet profile and measurements at the defined reference points, we calculate the expected average latency L_i, latency jitter J_i, and throughput T_i for each alternative a_i as follows:

$$L_i = p_{128} * L_{i,128} + p_{512} * L_{i,512} + p_{1500} * L_{i,1500} \qquad (3)$$

The expected average latency L_i describes the latency to be expected for the security mechanism and traffic profile under consideration.

$$J_i = p_{128} * J_{i,128} + p_{512} * J_{i,512} + p_{1500} * J_{i,1500} \qquad (4)$$

In contrast to a strict definition of latency jitter, Equation 4 does not use the maximum operator to determine an alternative's jitter. Instead, it allows the overall jitter assigned to alternative a_i to be influenced by the traffic profile. If only a small fraction of the expected traffic causes high values for jitter, this results in a better overall jitter value for the alternative. On the other hand, if nearly all traffic is part of a packet class with high jitter, the resulting overall jitter will be high, too. Therefore, by using the expected average latency jitter J_i, we try to reduce the overall jitter introduced by cryptographic mechanisms. If the strict definition of jitter was prefered, one would have to use the maximum operator for all packet sizes s with a probability $p_s \neq 0$.

$$\begin{gathered} T_i = \sum_s \frac{s * p_s}{\overline{sp}} * T_{i,s} \ , s = 128, 512, 1500 \\ \text{where } \overline{sp} = \sum_s s * p_s \end{gathered} \qquad (5)$$

As mentioned before, the traffic profile influences the achievable throughput. Equation 5 yields the throughput, which is to be expected for an alternative, i.e., a combination of the hardware used, the cryptographic mechanisms applied, and the traffic profile of the connection.

If there is more than one hardware or software option to perform cryptographic calculations for a specific security mechanism, we consider each option an alternative of its own. If, for example, the cryptographic operations for a specific TLS cipher suite can be performed on the CPU or a cryptographic accelerator card, this yields two candidates, which offer the same security properties but differ with regard to their QoS properties. Based on these estimations, for each parameter a QoS Utility is then calculated as described in Section V.

Packet type	Size [Byte]	Reference point [Byte]
Small	0 - 256	128
Medium	256 - 768	512
Large	>768	1500

TABLE II
TRAFFIC PROFILE

C. Energy Consumption

As the number of mobile devices is growing, energy consumption is becoming a critical aspect when applying security mechanisms. Therefore, additional energy consumed in order to provide security to users, should also be considered. Energy consumed to provide cryptographic services differs depending on the cryptographic primitives used. Further factors influencing the energy consumed for cryptographic operations are the hardware the calculations are performed on, and the implementation of the mechanisms—be it in hardware or in software. As energy consumption also depends on the packet size [10], we will use a traffic profile dependent calculation of the energy cost associated with an alternative. The adapters of ACCS therefore have to report the cost to apply an alternative's cryptographic mechanisms for each of the packet sizes defined in the traffic profile. Then an expected average energy consumption for alternative a_i under a traffic profile can be calculated by Equation 6.

$$E_i = p_{128} * E_{i,128} + p_{512} * E_{i,512} + p_{1500} * E_{i,1500} \qquad (6)$$

Currently, measurement of energy consumption is a non-trivial task due to the number of hardware components involved. A trivial possibility to include at least the energy consumed by the CPU consists in counting the cycles used to perform crypographic calculations. However, this does not reflect the actual cost associated with specific instructions. Today's CPUs often possess multiple performance counters, which enable a low-overhead on-chip measurement of a selection of instructions. These performance counters can also be used to obtain a more exact online energy estimation by counting energy-expensive instructions [11]. A prequisite to apply these energy counters is, that it is known how much energy is consumed for a specific instruction. If this is the case, we can measure the overhead in energy consumption caused by additional security mechanisms and additionally keep an account of the overall energy consumed for security mechanisms.

V. VALUE FUNCTIONS

By now, we have defined criteria which represent important aspects in choosing the most appropriate security protocol at hand. Subsequently, each alternative needs to be assigned a utility value with respect to each criterion expressing the attribute value's utility to the user. The criterion of effective bitstrength will be used as an example to demonstrate the application of value functions. We define a minimum and an optimum attribute value with the following meaning: Each alternative offering a smaller effective bitstrength than the minimum, must not be used. The minimum effective bitstrength is therefore used to filter out alternatives not fulfilling policy requirements. This is necessary to handle compensatory effects of MAUT: An alternative, violating one criterion, but scoring high regarding most other criteria, could achieve a higher rating than any other alternative and hence, would be selected if no prior filtering had been applied. Therefore, at least security-related criteria have to be used to filter out alternatives violating any single criterion. By setting the optimum effective bitstrength, the user indicates that a higher effective bitstrength may of course be used, but does not further increase the alternative's utility to the user in comparison to the optimal attribute value.

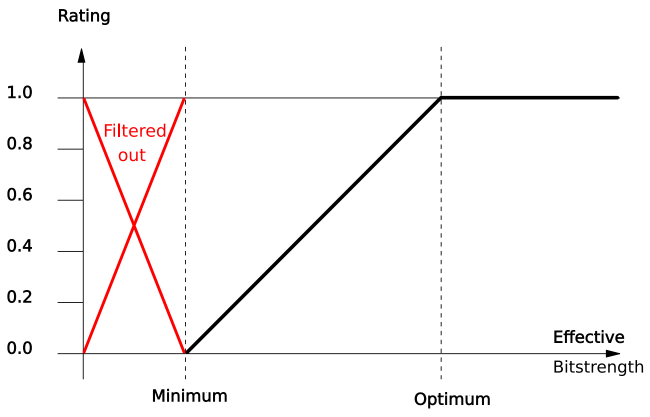

Fig. 4. Example of a value function: Effective Bitstrength

As the effective bitstrength is a security-related parameter, one might argue that a higher effective bitstrength is to be preferred, always. Although this may be true for some cases, e.g., attackers with large budgets and therefore large processing power, in general a user-dependent optimal value can be defined for which an assumed attacker with its constrained resources is supposed to be unable to carry out a successful attack. The decision between alternatives with an equal rating with regard to effective bitstrength is then dependent on the other criteria. Likewise, for every critierion an individual value function has to be defined. The example in Figure 4 uses a simple linear value function. To enable users to express their ideas of utility as exactly as possible also more complex functions, e.g., based on an exponential or logarithmic function could be used. Any value function has to map the attribute values to the interval $[0;1]$, with 1 indicating the highest utility to the user and 0 representing a permissible option but with no associated utility. To model a rapid decrease in utility, e.g., for increasing latency, the corresponding value function could be defined as depicted in Figure 5. To allow users, to also express indifference between ranges of attribute values, one can also define step functions, allowing them to set the range for which no distinction in rating is to be made. For reasons of transparency however, the simplest function, powerful enough

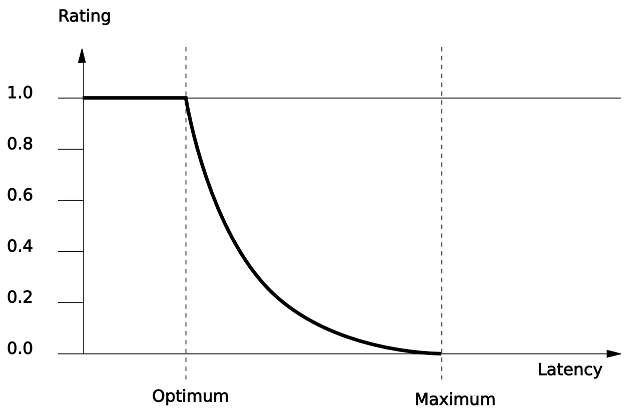

Fig. 5. Example of a value function: Latency

to express a user's intention, should be used to map attribute values to utility values.

VI. Aggregation

The last step in order to obtain a total ranking for each alternative consists in aggregating the now available utilities of the single criteria for each alternative. The weighted sum, we use for aggregation (see Equation 1), on the one hand, depends on the utility values of an alternative. On the other hand, the weights in front of each value function allow users to express the relative importance of criteria to them. A characteristic of MAUT aggregation methods, e.g., the weighted sum we use, is that in contrast to some other decision theory methods, they allow trade-offs between criteria. Those trade-offs are desirable in order to express, e.g., that one is willing to invest a certain amount of QoS in order to gain improved security properties. This characteristic of the MAUT however, requires us to filter out alternatives which do not fulfill necessary requirements. If a user demands at least partial forward secrecy, then any alternative offering no forward secrecy at all is to be dropped from further processing. Otherwise, due to the compensatory effects of MAUT, it could get a higher rating than any other policy-conforming alternatives, e.g., because it offers excellent QoS properties.

The determination of weights should be done carefully, as these—besides the parametrization of the value functions—are an important factor in calibrating the MAUT. One possibility is to just estimate the weights. A more analytic approach to determine the weights is described in the Analytic Hierarchy Process. In the AHP, criteria of the same hierarchy layer with the same parent, are compared pairwise on a simple discrete scale and the resulting values are entered in a quadratic matrix. The eigenvector method, used to calculate the weights based on the obtained matrix, enables analytical detection of inconsistencies in a user's preferences. For this reason, in decision theory this method is often preferred over simpler methods to determine weights.

When configuring the system for a group of users, a sensitivity analysis for the chosen parameters, i.e. the parametrization of functions and the determined weights, should be performed on a some predefined sets of alternatives, realistic in the given scenario, to verify the intended behavior.

The result of the aggregation is a utility value for each candidate, which can be used to order the candidates and choose the best candidate—the one with the highest utility.

VII. Related Work

[12] modeled Security as a QoS parameter. The strength of cryptographic algorithms, the key length, and a few other parameters are examined; however, the authors do not compare and choose different protocols or mechanisms.

Also looking at security parameters is [13]. The authors point out that "security requirements are volatile and change frequently". The major difference is that our approach does not rely on a specific protocol to jointly negotiate the use of a security protocol but automatically uses existing protocols without an additional negotiation. Our solution also supports requirements besides security.

The authors of [14] looked into the tradeoff of energy consumption and security for small wireless devices. Their approach is based in a simple classification of packets. Packets may require high, medium, or low security. Also management, control, and data packets are treated differently. The security is just represented by the years the encryption should be secure and a probability that the message authentication is broken. This is calculated by a formula using todays estimations. Basically, the user can only choose between a few different ciphers and key lengths. We provide a more flexible approach and allow ways to integrate known attacks into the security estimation, allow for different connections to have different requirements, and are able to take multiple criteria into consideration.

VIII. Conclusion and Future Work

In this paper, we have presented our policy-controlled, automated approach for determining the security protocol which best complies with a user's intentions. We do not only support different security protocols and cipher suites but also presented how to examine the supported cryptographic algorithms. A major advantage of our rating approach is the modeling and inclusion of QoS and energy consumption aspects in the rating of individual security protocols and cipher suites.

The next possible step is adding further aspects, like Mobility and Robustness. To do this, the exact influence of the security protocols has to be examined and modeled. But also additional criteria of the existing aspects can be added and evaluated, e.g., data volume and key lifetime.

In addition, further evaluation work on the effects of the ω-parameters, possibly with different users, may yield additional insights into the approach.

Acknowledgments

The authors would like to thank Dr. Christoph Sorge for valuable input and discussions.

References

[1] A. Whitten and J. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0", *8th USENIX Security Symposium*, 1999.

[2] P. Gutmann and I. Grigg, "Security usability", *IEEE Security and Privacy*, vol. 3, no. 4, pp. 56–58, 2005.

[3] L. Völker, M. Schöller, and M. Zitterbart, "Introducing QoS mechanisms into the IPsec packet processing", in *Proc. 32nd IEEE Conference on Local Computer Networks LCN 2007*, 15–18 Oct. 2007, pp. 360–367.

[4] T. Saaty, *Fundamentals of the Analytic Hierarchy Process.* RWS Publications, 4922 Ellsworth Avenue, Pittsburgh, PA 15413, 2000.

[5] C. Hwang and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications.* Berlin, Heidelberg, New York: Springer Verlag, 1981.

[6] J. Figueira, V. Mousseau, and B. Roy, "ELECTRE methods", in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Boston, Dordrecht, London: Springer Verlag, 2005, pp. 133–162.

[7] R. Keeney and H. Raiffa, *Decisions with multiple objectives: Preferences and value tradeoffs.* J. Wiley, New York, 1976.

[8] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management", National Institute of Standards and Technology (NIST), NIST Special Publication 800-57, revised, Mar 2007, see http://csrc.nist.gov/publications/nistpubs/index.html.

[9] A. D. Keromytis, J. L. Wright, T. D. Raadt, and M. Burnside, "Cryptography as an operating system service: A case study", *ACM Trans. Comput. Syst.*, vol. 24, no. 1, pp. 1–38, 2006.

[10] P. Keeratiwintakorn and P. Krishnamurthy, "Energy efficient security services for limited wireless devices", *Wireless Pervasive Computing, 2006 1st International Symposium on*, pp. 1–6, 16-18 Jan. 2006.

[11] A. Weissel and F. Bellosa, "Process cruise control: event-driven clock scaling for dynamic power management", in *CASES '02: Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems.* New York, NY, USA: ACM, 2002, pp. 238–246.

[12] C. Irvine and T. Levin, "Quality of Security Service", in *NSPW '00: Proceedings of the 2000 workshop on New security paradigms.* New York, NY, USA: ACM, 2000, pp. 91–99.

[13] A. Klenk, M. Masekowsky, H. Niedermayer, and G. Carle, "ESAF – an Extensible Security Adaptation Framework", in *Proceedings of the 10th Nordic Workshop on Secure IT-systems (NordSec05), Tartu Estonia*, Oct. 2005.

[14] P. Keeratiwintakorn and P. Krishnamurthy, "Energy efficient security services for limited wireless devices", in *Proc. 1st International Symposium on Wireless Pervasive Computing*, 16–18 Jan. 2006, pp. 1–6.

[15] S. Kent and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301 (Proposed Standard), Dec. 2005.

[16] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346 (Proposed Standard), Apr. 2006, updated by RFCs 4366, 4680, 4681.