# A Method for Describing Structure of System Security Based on Trust and Authentication

TSUKASA MAEDA and MASAHITO KURIHARA
Graduate School of Information Science & Technology
Hokkaido University
Sapporo, Japan
tsukasa.maeda@rsa.com, kurihara@ist.hokudai.ac.jp

*Abstract*—**In this paper, we propose a method by which frontline engineers in system development fields can readily describe the structure of the security of systems. This method, based on the assumption of the use of standard encryption technologies and existing cryptographic protocols, reveals hidden security threats and vulnerabilities of systems. It extracts only security elements that constitute the trust relationship of system components, describing the relation between the elements, and analyzing the relation. This method provides a valuable assistance tool to build secure systems, because it works as an efficient communication paradigm between stakeholders of a system to help them in understanding the security of the system and confirming that their security requirements are fulfilled.**

***Keywords-system description; system security; authentication***

## I. INTRODUCTION

As distributed computing is becoming a general architecture of information systems, security becomes a mandatory requirement of such systems. As the concern about security increases, re-evaluation of security is currently underway in many existing information systems. And it becomes a common practice to put security properties into the requirements of systems and validate their adequacy during the system development process. Responding to this trend, many security technologies and products providing them have been developed and become available. Also, many research activities and standardization efforts on the formal specification and evaluation of security properties for system development are going on now [1][2][3].

However, despite these efforts, the number of security incidents in information systems has been increasing rapidly. While new attacks like phishing and malware have been developed and damage caused by them has spread extensively, the development of countermeasures against such attacks has fallen behind, and field engineers are hard-pressed to keep pace by applying ad hoc, daily-basis measures to their systems.

The reason for the difficulty in developing secure systems is twofold. Firstly, it is difficult to discover threats and vulnerabilities hidden behind the complex structure of a system. This means not only that it is difficult to explore the weakness of the security of a system consisting of many components, but also that the aggregation of individual security measures does not necessarily make the entire system more secure. Furthermore, though vulnerabilities envisioned from past experiences and incidents can be addressed relatively easily, it is very difficult to detect unknown new weaknesses or prescribe measures against them. Secondly, there is the difficulty of communications between stakeholders of a system. Because there is no universally established precise and easy way of describing the security of the system as provided by the security measures designed, it is very hard for stakeholders of a system to understand the security of the system, confirm that their requirements have been satisfied, and agree on the security specification of the system.

In this paper, we provide the framework of an expressive general-purpose method of describing the structure of the security of systems to solve these difficulties. With this method, frontline engineers in system development fields can easily verify that the design of the system will satisfy the security requirements. It assists the engineers in checking if any hidden defects in the security exist in the system. It also may be used as a communication tool between stakeholders of a system by providing them with clear and accurate information about the security structure of the system for both development and security assessment.

The security properties studied in this paper are “confidentiality”, “integrity”, and “authenticity” - security properties as defined in ISO27002 [4]. These properties are capable of being delivered through the use of cryptographic technologies. Storing information safely and exchanging it securely with only trusted parties are fundamentals of the confidentiality, integrity, and authenticity properties of systems.

To accomplish the purposes addressed above, the description method we introduce has the following features. For resolving the difficulty of identifying the weakness(es) of a system, the method does not describe the security of the system by identifying individual threats and vulnerabilities against it, as is the more typical method of the security description, but represents these threats and vulnerabilities as the defects of *structure* of the system’s security. For providing a clear and easy-to-understand description, the method extracts the security elements constituting the trust relationship between the components of a system, and only uses them to describe the structure of the security of the system. By doing so, vulnerabilities existing in the system

appear automatically without being hidden behind the complexity of the implementation of the system. Also for the purpose, the method avoids using detailed security primitives related to encryption technologies. Rather, it uses abstract security service features realized by existing cryptographic protocols such as SSL/TLS [5] and Kerberos [6] to describe the security of systems.

The rest of this paper is organized as follows. In Section II, we define the components being used to describe the security of a system. In Section III, we describe the steps involved in describing the security of a system. In Section IV, we examine the expressive power of the description method. Finally in Section V, we review the rationale of the description method and discuss the relationship of the description method to other research studies on the security analysis of systems.

## II. THE DESCRIPTION METHOD

### A. Building Blocks

#### 1) Entity

A system is a set of entities. That is, system = $\{e_1, e_2, ..., e_k\}$ where each $e_i$ is an entity.

An entity $e$ is a 6-tuple (Fig. 1) containing an IDENTITY, a SECRET, a CREDENTIAL, an EXECUTION, an ADJ, and a TRUST as defined below.

IDENTITY is a piece of information that uniquely identifies an entity within the system. It could be the unique name given to the entity or the address or any other information that can be used to recognize the entity uniquely. Hereafter, IDENTITY is used to specify an entity.

SECRET is a set of confidential items $s$ being used to authenticate an entity. SECRET may have two or more elements. An entity is said to be authenticated when its possession of SECRET elements is verified. Private keys, symmetric keys, passwords, and PIN numbers are typical examples of SECRET elements. A SECRET element has a strength attribute. The strength of the SECRET element correlates to the amount of work needed to break the algorithm using the SECRET element as the key or to determine the value of the SECRET element, and is expressed in terms of the length of the comparable symmetric key that requires the same amount of work to break the key. Let $\perp$ be a SECRET element with 0 strength. An entity having only $\perp$ may use its IDENTITY to be authenticated. However, because IDENTITY is public information, using IDENTITY for authentication does not provide any significant security to the system.

The CREDENTIAL of an entity is a set of pieces of information that are requested for presentation by other entities to authenticate the entity. Elements of the CREDENTIAL are outputs of processes ($\phi$) that take SECRET elements as inputs and generate outputs used for authentication. The form of the outputs varies according to the inputs. For example, the output could be a message with a digital signature if a private key is input, encrypted data if the input is a symmetric key, and the same password or more likely its hash value if a password is input.

| | |
|---|---|
| *IDENTITY*: | $i$ |
| *SECRET*: | $\{s_b\}, b = 1, \cdots, m$ |
| *CREDENTIAL*: | $\{\phi_b(s_b, \ldots)\}, b = 1, \cdots, m$ |
| *EXECUTION* | |
| *ADJ*: | $e_1, \cdots, e_p$ |
| *TRUST*: | $\{(e_c, \gamma_{e_c}(t_{e_c}, \ldots))\} c = 1, \cdots, l$ |

Figure 1. Entity

EXECUTION defines the operational behavior of the entity. For example, it may be a set of traces or actual program code.[1]

ADJ is a set of IDENTITYs of entities that are directly adjacent to the entity. [2]

The TRUST of an entity $i$ is a set of pairs, each consisting of the IDENTITY of a trusted entity $e$ the trusting entity $i$ has decided to trust and a verification process $\gamma_e(t_e,\ldots)$. The verification process receives one of the CREDENTIAL elements of the entity being trusted, say, $e$ and evaluates the correctness of that CREDENTIAL element. $t_e$ is a piece of information that the trusting entity $i$ uses to verify $e$'s possession of the SECRET element $s$. If $s$ is a symmetric key for example, $t$ is the same key. If $s$ is a private key, $t$ would be the public key pairing with $s$.

An entity trusts other entities if the entity believes that the trusted entities operate as specified. In this description method especially, it means that the trusted entity protects information that it stores, maintains the confidentiality and integrity of the information, and has appropriate access control, so that it can communicate with only authenticated entities.

#### 2) Type of entity

An entity is an abstract object being defined by extracting the trust attributes mentioned in Section II.A.1 from an actual component of a system. Entities are classified into two types: execution and link (Fig. 2). An execution entity is an object that performs information processing while interacting with other execution entities. Computer programs, services and users are typical examples of execution entities. A link entity is a virtual entity that models a communication channel established by a cryptographic protocol such as SSL/TLS and Kerberos between two interacting execution entities. We may consider a link entity as a First-In First-Out two-way buffering process.

#### 3) Link entity

As its components, a link entity presents typical security services that general cryptographic protocols provide to a system: authentication of interacting entities, session key exchange, and message encryption. Components of a link entity are configured as follows (Fig. 3).

---

[1] In this paper, we usually omit this from diagrams for simplicity.

[2] We often omit ADJ from diagrams if its elements are obvious.

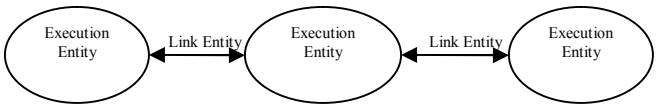


Figure 2. System

IDENTITY could be any piece of information that can specify the link entity uniquely. For instance, the session ID, or simply the name given to the entity could constitute its IDENTITY.

A session key becomes the element of SECRET. Only the session key can distinguish messages in the session from other sessions.

Messages encrypted with the session key Ks can be used as the elements of CREDENTIAL because execution entities on both sides can confirm that they are using a proper communication channel by encrypting and successfully decrypting messages.

EXECUTION consists of session establishment (authentication and key exchange) and encrypted message exchanges.

TRUST borrows elements from the TRUSTs of the execution entities on both sides of the session. These elements that were used for authentication upon establishment of the session are copied to the TRUST. If it is a session created by mutual authentication, the IDENTITY and TRUST element $(e, \gamma_e(t_e,\ldots))$ from both sides are copied. In the case of unilateral authentication, $(e, \gamma_e(t_e,\ldots))$ from the authenticating entity and $(e,\perp)$ from the authenticated side are copied.

The IDENTITYs of the two communicating execution entities that generated the session are set as the ADJ elements.

Since a link entity is created as the result of the authentication by the execution entities on both sides of the link, the elements of the TRUSTs of the execution entities used for the authentication are copied, and used to express the trust relation between the link entity and the execution entities.

While the link entity is configured as mentioned above, the evaluation process verifying the element of the CREDENTIAL of the link entity by checking if $E(m)_{Ks}$ is encrypted/decrypted correctly, is added to the TRUSTs of the execution entities of both sides. For the execution entities, a trustworthy link entity is a communication channel which assures secure message exchange with an authenticated entity on the other side.

| | |
|---|---|
| *IDENTITY*: | *n* |
| *SECRET*: | Session Key *Ks* |
| *CREDENTIAL*: | $\phi(Ks, m) = E(m)_{Ks}$ |
| *TRUST*: | From B $(e_a, \gamma_a(t_a,\ldots))$, From A $(e_b, \gamma_b(t_b,\ldots))$ |
| *We omit EXECUTION and ADJ here for simplicity* | |

Figure 3. link entity

### B. *Description Method Rule (Entity Combination Rule)*

Provided that they are authenticated mutually with SECRET elements with comparable strength, two entities that are directly adjacent to (and trusted by) each other can combine to form a single entity thus concealing the communication between them. The communication process is deemed an internal processing in this situation. Within the combination, some elements of SECRET, CREDENTIAL and TRUST may be shared if certain conditions are satisfied.

The adequacy and comparability of the strength of the SECRET elements varies application by application and has to be judged in accordance with the overall system requirements.

Entity A authenticates entity B by validating the element of B's CREDENTIAL that it presents, using the validation process defined in A's TRUST. To avoid replay attacks, validation must include a test of the freshness or currency of the CREDENTIAL element presented, meaning that it cannot be an imitation of an older element.

Sharing of various elements becomes possible only when such sharing could actually happen in application processes. For example, to share a CREDENTIAL element of one entity, the element actually has to be forwarded to another (combining partner) entity and become available for presentation to other outside entities. To make a TRUST element shared, either transferring the validation process or delegation of such a function must actually be possible.

Conditions indicating the processes that must occur for successful sharing of elements, if any, must be noted in the Sharing Conditions of the description diagram to avoid deviation between the actual security structure and the one described.

## III. THE PROCESS OF SYSTEM DESCRIPTION

### A. *Steps and Interpretations of the Description*

The structure of the security of a system is described in the following order.

- Identifying execution entities in the system and diagramming them in a chart.
- Determining the SECRET, CREDENTIAL and TRUST elements of the execution entities
- Specifying the link entities
- Configuring the link entities
- Applying the combination rule;

while (there are two entities that can be combined) do
{combine the entities; share elements if possible}

We show an example of the steps describing the structure of the security of a Web application system using SSL/TLS server side authentication in the following.

*1) Identifying execution entities in the system*

We specify components of the system such as users and programs/processes that have the capability of satisfying the entity security requirements as mentioned in Section II.A.1. We then derive execution entities corresponding to each of these components. If there are communication links between these components, we identify them here.

We show the browser (B) in the Web application, the Web server (C) and the user (A) as entities in Fig. 4. The horizontal line expresses the SSL/TLS channel. The EXECUTION and ADJ of the entities are omitted. The CREDENTIAL is presented when necessary.

*2) Determining the SECRET, CREDENTIAL and TRUST of the execution entities*

For every execution entity identified in Section III.A.1, we identify and specify the SECRET, CREDENTIAL and TRUST of the entity. Firstly, we specify the SECRET elements that are used to authenticate the entities. Secondly, the CREDENTIAL elements of the entities are specified. They are outputs of processes, and are intended to be given to other entities for authentication. In the example, however, because the relationship between the SECRET elements and the CREDENTIAL elements are obvious, the description of the CREDENTIAL is omitted. Finally, pairs of IDENTITYs of entities that the entity trusts, and the validation processes that validate their CREDENTIAL elements, are copied into the TRUST of the entity. The validation processes may not be simple functions that merely input CREDENTIAL elements and return the result of validation. They might be more complex interacting processes. For example, validation processes may include a series of challenge-response actions: (1) the validation side generates a random nonce and sends it to the entity being authenticated, (2) the sent nonce is converted with a SECRET element and sent back for validation. In the case of the SSL/TLS protocol, the client generates a random number during its session key generation process, encrypts it with the server's public key and sends it to the server. The client can authenticate the server when the session key is successfully generated and agreed.

We show the elements of these sets for the Web application example in Fig. 5. For simplicity, an element of TRUST (Y, $\gamma_Y(t_Y,\ldots)$) of entity X with entity Y is denoted by $T_{X,Y}(t_Y,\ldots)$. The elements shown in the diagram mean;

- PW: the user's password
- Ks, Kp: server's private key and public key pair
- $T_{A.B}(\perp)$: the user authenticates the browser because it is there (exists)
- $T_{A.C}$ ($\perp$): the user authenticates the server with its name (i.e. URL)
- $T_{B.A}$ ($\perp$): the browser only receives an input from the user.
- $T_{B.C}$ (Kp, $r$): the browser authenticates the server with the server's public key. $r$ means that a random number $r$ is also involved in the validation.
- $T_{C.A}$ (PW): the server authenticates the user with the user's password.
- $T_{C.B}$ ($\perp$): the server identifies the browser with the IP address of the user's computer and the browser's process ID.

| *IDENTITY* | A | B | C |
|---|---|---|---|
| *SECRET* | | | |
| *TRUST* | | | |

Figure 4. Specifying execution entities

| A | B | C |
|---|---|---|
| PW | $\perp$ | Ks |
| $T_{A.B}(\perp)$<br>$T_{A.C}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.C}(Kp,r)$ | $T_{C.A}(PW)$<br>$T_{C.B}(\perp)$ |

Figure 5. Determining SECRET, CREDENTIAL and TRUST of the execution entities

*3) Specifying link entities*

If communication channels exist between system components corresponding to execution entities, link entities are inserted there and named with an arbitrary IDENTITY used to identify the entity uniquely. The link entity corresponding to the SSL/TLS channel in the above example is named D in Fig. 6.

*4) Configuring the link entities*

Components of the link entity are configured as follows. The session key Kd generated during the session negotiation of the cryptographic protocol is set as the SECRET element of the entity. Both the element $T_{B.C}(Kp,r)$ in the TRUST of entity B and the element $T_{C.B}(\perp)$ in the TRUST of entity C, used to authenticate C and B respectively and establish link D, are copied into the TRUST of link entity D. To the TRUSTs of B and C, $T_{B.D}(Kd)$ and $T_{C.D}(Kd)$ are added respectively as elements by which D's possession of Kd may be verified. Fig. 7 shows the result of the configuration.

- Kd: the SSL/TLS session key
- $T_{B.D}(Kd)$: Browser verification of decryption of a message with the session key Kd
- $T_{C.D}(Kd)$: Server verification of decryption of a message with the session key Kd

*5) Applying the combination rule*

If two entities directly adjacent to each other have a relation of mutual authentication, they are combined. In the example, C has the SECRET element Ks which is verifiable by D with the public key Kp. And D has Kd which is verifiable by C with the symmetric key Kd. A random number $r$ is used in the verification process by D to check the freshness of C's CREDENTIAL element being given to D. Kd is a short-live session key. So, C and D can authenticate each other without risk of replay of the CREDENTIAL elements. Fig. 8 shows the relationship of mutual authentication between C and D, and elements that may be deleted after the combination.

| A | B | D | C |
|---|---|---|---|
| PW | $\perp$ | | Ks |
| $T_{A.B}(\perp)$<br>$T_{A.C}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.C}(Kp,r)$ | | $T_{C.A}(PW)$<br>$T_{C.B}(\perp)$ |

Figure 6. Specifying link entities

| A | B | D | C |
|---|---|---|---|
| PW | $\perp$ ⋯▶ | **Kd** | ◀⋯ Ks |
| $T_{A.B}(\perp)$<br>$T_{A.C}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.C}(Kp,r)$<br>**$T_{B.D}(Kd)$** | **$T_{D.C}(Kp,r)$**<br>**$T_{D.B}(\perp)$** ◀ | $T_{C.A}(PW)$<br>$T_{C.B}(\perp)$<br>**$T_{C.D}(Kd)$** |

Figure 7. Configuration of the link entity

| A | B | D | C |
|---|---|---|---|
| PW | ⊥ | **Kd** | Ks |
| $T_{A.B}(\perp)$<br>$T_{A.C}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.C}(Kp,r)$<br>**$T_{B.D}$(Kd)** | ~~$T_{D.C}$(Kp,r)~~<br>**$T_{D.B}(\perp)$** | $T_{C.A}$(PW)<br>$T_{C.B}(\perp)$<br>~~$T_{C.D}$ (Kd)~~ |

Figure 8. Applying the combination rule

As a result of the combination, $T_{C.D}$(Kd) and $T_{D.C}$(Kp) that were used for the mutual authentication may be removed (obscured) from the structure of the security of the system without weakening it. As D is just a communication channel, we may consider that C comes closer to B through this combination. We name the combined entity E.

Because it was D that had a direct interface with B before the combination, all elements of C that are supposed to be used as elements of E after the combination to express the trust relationship with B, must actually be available for D. In the example, because D is just a communication channel and C is taking care of all operations regarding the trust relation, i.e. giving the CREDENTIAL element to B and verifying B's CREDENTIAL element, all of these elements can be shared and become elements of E automatically. We do not have to specify any sharing conditions on this combination (Fig. 9).

## IV. Power of the Description Method

### A. Description of System Threats and Vulnerabilities

One of the distinguishing features of our description method is its capability of representing threats (attacks) and vulnerabilities (weaknesses) of a system as structures of trust relationship between the system components.

Since our description only specifies the trust relationship between "legitimate" entities, no entities are explicitly introduced for representing "attackers". Threats that attackers would bring to the system, emerge as attempts by attackers to replace legitimate entities with fake entities controlled by them. Attackers attempt to get access to data protected by valid entities, by attacking the defects in trust relationships between the entities. If such replacement is made undetected by other entities, attacks can succeed, and security is broken. Vulnerabilities of a system translate to replaceable entities.

The replacement of entities can succeed if either internal behaviors of entities are mimicked or their appearance is copied. Internal behaviors can be mimicked if elements of either SECRETs or CREDENTIALs are broken, meaning SECRET elements do not have sufficient strength, or there are defects in the processes generating CREDENTIAL elements.

| A | B | E |
|---|---|---|
| PW | ⊥ | Ks,Kd |
| $T_{A.B}(\perp)$<br>$T_{A.E}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.E}(Kp,r)$<br>$T_{B.E}$(Kd) | $T_{E.A}$(PW)<br>$T_{E.B}(\perp)$ |
| *Sharing Condition* | | No |

Figure 9. Structure of the security of the Web application

In the following, we show how some known security threats can be manifested in terms of replacement of entities.

- Eavesdropping and data forgery: attacks on the network can be instantiated as replacement of link entities.
- Phishing: phishing is an attack being carried out outside the system. It is hard to distinguish a legitimate password from a stolen one after it has been stolen. It is important to consider measures to prevent password theft.
- Pharming: replacement of a server-side entity
- MITM (Man in the Middle): replacement of an intermediate entity that corresponds with the browser, for example.
- Guessing attack: replacement of execution entities because of inadequate strength of SECRET elements in the password type.
- Hijack: replacement of a link entity because of inadequate strength of a SECRET element (i.e. the session key)
- Replay: replacement of an entity by imitating the appearance of the entity with pre-recorded behaviors.

### B. Example of Description of Systems

We confirm the expressive power of the description method through several authentication technologies.

#### 1) Kerberos

Kerberos is an authentication protocol for user authentication in UNIX and Windows systems when users try to access system resources on the network. Kerberos is a cryptographic protocol that is not a subject to be analyzed in this paper, but is rather an existing authentication method to form an illustrative link entity. However, because Kerberos can be regarded as a system consisting of a client and several servers providing security services, it can also be a good example of a system providing a security service. It consists of several entities interacting with each other via cryptographic protocols.

Kerberos consists of the client terminal (C) through which the user accesses network resources, the authentication server that verifies the user's password (AS), the ticket granting server that issues a ticket to allow the user to access network resources after authentication (TGS) and the actual application server that the user wants to access (S).

In the Kerberos specification, trust relationships between the components were established with symmetric keys previously as follows; Kc between C and AS, Ktgs between AS and TGS and Ks between TGS and S. These keys become the SECRET elements of the components and therefore, inputs of the elements of the CREDENTIALs.

Fig.10 shows the formation of the components of Kerberos. The circles specify the components that become the execution entities. Lines are communication paths that become link entities.

In the specification of Kerberos, Kc is not a SECRET element for C, but the encryption key derived from the user's password PW in C. AS authenticates the user with Kc. C does not have its own SECRET element, and has no trust

relationship with other components at all. That is to say, C is a replaceable entity because no other entities authenticate C. Also, the user is not a part of the specification for Kerberos. If we want to use Kerberos for the user authentication service of a system, it is important to consider implementing a secure PW input method and a valid authentication scheme for the client terminal. This is not a problem solely for Kerberos, but a common issue of standard authentication technologies available today. A major cause of security threats brought by malware is this vulnerability. Also, it must be reiterated that the strength of Kc is no more than the strength of PW because Kc is derived from the PW. It should usually be viewed as being weak.

For simplicity, hereafter we assume that a certain secure trust relationship between the user and C has already been established. We analyze the security structure of Kerberos within the scope of its specification.

Between AS and C, a message *m* encrypted with Kc is sent from AS to C only once in one direction. C can confirm that AS has Kc because C can decrypt the encrypted *m* sent from AS with Kc. Also because *m* contains a timestamp, C can verify that *m* is not a replayed message output from AS previously. By contrast, by encrypting *m* with Kc, AS can assure itself that *m* is only available to C, because only C can decrypt *m*. Since C does not send a message back, we do not have to worry that C is being mimicked.

Kc is set as the SECRET element of the link entity, named L1, between AS and C. The TRUST elements of C and AS used on each other are copied to TRUST for L1. As a result of this L1 configuration, C, L1 and AS can be combined. We name the combined entity D. Fig. 11 shows the trust relationships of these entities. CREDENTIALs are omitted from this example. Here, $T_{X.Y}$ (Kc, *m*, *t*) means that m contains a timestamp *t*.

The message *m* from AS to C contains a session key Kt that was generated by AS for communication between C and TGS, and an encrypted Kt using Ktgs as well. The combined entity D contains Kt and is able to send the encrypted Kt to the TGS. This means that a part of the CREDENTIAL generation processes for AS is shared in D through the combination that created D.

Between D and TGS, the encrypted Kt, is sent from D to TGS. TGS decrypts the encrypted Kt using Ktgs that is shared with AS, and sends a necessary message back to D securely using Kt as the session key. We can regard this message exchange as a protocol with which D and TGS can mutually authenticate with the shared Ktgs and generate the session key Kt. D uses $E(Kt)_{Ktgs}$, that D borrowed from AS through the combination, as its element of CREDENTIAL and TRUST with TGS. The trust relationship among D, TGS and the link entity L2 are illustrated in Fig. 12.

We can confirm the freshness of D via a timestamp contained in the message sent from D to TGS. Because Kt is a (short-live) session key, we can also confirm that TGS is not mimicked if D can decrypt the message successfully. With this mutual authentication, D, L2 and TGS may be combined.

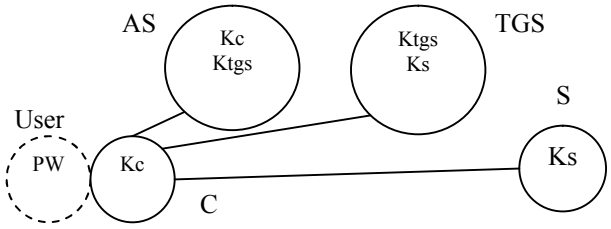


Figure 10. Kerberos configuration

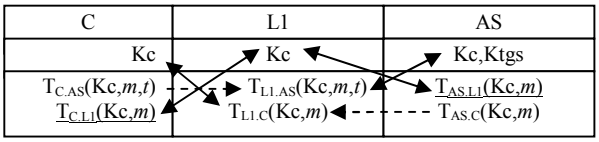


Figure 11. Kerberos: Combination of C, L1 and AS

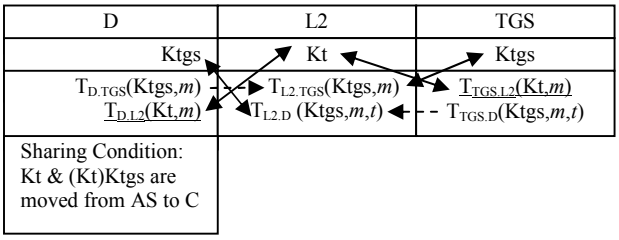


Figure 12. Kerberos: Combination of E, L2 and TGS

Throughout the Kerberos specification, this combined entity establishes a secure communication to the ultimate application server S. At the end of the protocol, we can combine D, AS, TGS and S into a single entity. As a result, with one combined entity containing all components of Kerberos, we can conclude that we have a secure communication channel between the user and S using Kerberos.

*2) SSL / TLS server authentication*

We investigate the SSL/TLS example presented in Section III.A in more detail. As mentioned in Section IV.A, the possibility of replacing entities of a system, that is, the vulnerabilities of the system, is due to either inadequate strength of the SECRET elements, defects in the trust relationships between entities, or the possibility of imitation of instances of entities. It is possible for us to check if a vulnerability exists in a system through the use of trust relationship diagrams in this description method by following the steps below:

*a) Inadequate strength of SECRET elements:*

Check the existence of SECRET elements. Calculate the strength of SECRET elements using, for example, the NIST SP800-57 [7] guideline if any. The minimum value of these strengths becomes the overall authentication strength of the entity

*b) Defects in trust relationships:*

Confirm that CREDENTIAL elements for every entity are referred to, and verified, in the TRUST of any other trusting entity. If not, the entity is replaceable. If it is a case, check if the IDENTITY of the verifying entity is in ADJ of the verified entity. If it is not the case at all, the verified

entity is replaceable unless there is a secure way of transferring CREDENTIAL elements of the verified entity to the verifying entities.

*c) Possibility of imitation of instances:*

Check if for every entity CREDENTIAL elements $\phi$ contain any input that varies over time (for example, a timestamp, an expiration date, a random nonce, or an aggregated count), and are verified by other entities. If no CREDENTIAL elements have such input, the entity is replaceable.

By applying these steps to the Web application diagram presented in Fig. 9, we can see that the system has several vulnerabilities (replaceable entities) as follows.

For A:

- The strength of the PW may be weak. It is common understanding in the industry that the entropy of a password is at most 60 bits [8]. It raises the risk of spoofing. The password can be stolen through a dictionary attack.
- Since $T_{E.A}$ (PW), which is the only TRUST element that verifies PW, does not have any input that varies over time, an attacker can mimic the behavior of A by merely presenting the PW that was previously stolen by the attacker pretending to be B. A could become a good target for a phishing attack.
- Because the entity E that is the only entity trying to authenticate A with PW, is not adjacent to A directly, and B is replaceable, a MITM attack is possible.

For B:

- B does not have any SECRET elements, and all other entities that have TRUST elements with B use $\perp$ to verify B. Therefore, B is a replaceable entity. All information passing through B can be stolen. MITM is easily carried out.

For E:

- Though E is authenticated by B, because B is a replaceable entity, B and E are replaceable together from A's perspective. Farming is a possible attack.

These analyses explain vulnerabilities of Web applications we are experiencing today.

For a further investigation of the security of the Web application, let's change the user authentication method from simple PW to use of a onetime password. As the CREDENTIAL element of A given to E, we use a composite onetime password generation process $\phi$(PW, $t$) inputting PW and the time t instead of a static PW. The diagram of the trust relationship becomes Fig. 13.

| | A | B | E |
|---|---|---|---|
| | PW | $\perp$ | Ks,Kd |
| *CREDENTIAL* | $\phi$(PW,$t$) | $\perp$ | Omitted |
| | $T_{A.B}(\perp)$<br>$T_{A.E}(\perp)$ | $T_{B.A}(\perp)$<br>$T_{B.E}$(Kp,$r$)<br>$T_{B.E}$(Kd) | $T_{E.A}$(PW)<br>$T_{E.B}(\perp)$ |
| | *Sharing Condition* | | No |

Figure 13. The Trust relationship with onetime password

Through this modification, the weakness of A coming from the inadequate strength of the SECTRET element, and the possibility of imitation are eliminated. Only one problem remains.

- Because the entity E that is the only entity trying to authenticate A with PW, is not adjacent to A directly, and B is replaceable, a MITM attack is possible.

Because the vulnerabilities of B and E will not change, we can easily understand that the combination of a onetime password scheme and SSL/TLS server-side authentication can improve the security strength of authenticating A; however this does not solve the problem of a MITM attack implemented by hijacking the browser or misdirecting the user to a faulty site. This clearly shows the limitation of the onetime password scheme that is sometimes difficult to recognize.

## V. DISCUSSION

### A. Trust Elements - Configuration of the entity

#### 1) SECRET

In developing systems with mobile equipment and/or embedded devices, it has been general practice to include a device identification and authentication scheme into the device's system design. When designing internal systems, however, that consist of computing equipment continuously interconnected via a corporate network, it is often the case that no authentication means is considered; devices are trusted when they join the network. The increase in attacks by insiders and incidents of malware indicate that such a trusting assumption is no longer good practice. In addition, as devices with no innate consideration for being authenticated, such as PCs, become major components of a system, it becomes difficult, if not impossible, to maintain the security of the system without exchanging CREDENTIAL elements among system components to verify trust relationships. Based on this understanding, we stipulate the possession of SECRET elements as the basis of the security of systems, and build upon this foundation.

#### 2) Link entity

This description method provides a means to analyze the security of systems on the premise of standard, proven encryption technologies and existing well-designed cryptographic protocols that enable security services for mutual authentication, key exchange and message encryption. When we design secure systems, it is common to use existing cryptographic protocols such as SSL/TLS, Kerberos and IPSec as tools to implement needed security services, or sometimes to use custom protocols specifically designed by a security expert. The premise above is well suited to this situation. Traditional development engineers are trained to use security protocols, not to design them. They may make use of this description method to analyze the security of the systems they deploy and communicate with stakeholders of the system to agree on the system design.

It is important, however, for engineers to design systems so that cryptographic protocols used in the systems can be replaced easily when vulnerabilities are found in the protocols. Even a well-studied cryptographic protocol could

have vulnerabilities. For example, Kerberos was revised to V5 when some weaknesses were recognized in V4 [9].

### B. Related Work

In the 1970's, the foundational results of years of cryptographic research emerged, and established cryptographic standards in the Internet age. Since the late 1970's through the 1990's, research into the analysis of system security became active and flourished based on these results. In particular, the application of formal methods to cryptographic protocols has been a mainstay of the research activities. The first significant work in the area was done by Dolev and Yao[10]. Though the research did not go further for several reasons, it included a powerful intruder model upon which many later works in this area are based. After Dolev and Yao, perhaps the most influential paper in this field was published by Burrows, Abadi and Needham[11]. Their logic, called BAN logic, starts with an initial set of beliefs on security properties of entities based on the premise of correctness of encryption schemes. It then builds on this premise with statements about messages exchanged throughout the course of a protocol between the entities, and concludes by verifying the correctness of the protocol. The description method introduced in this paper starts with trust relationships among entities that can be assumed by the security properties derived from the correct use of encryption technologies and cryptographic protocols that have been studied in the prior art. It builds upon these assumptions with a description of the structure of the security of a system. Other approaches to formal cryptographic protocol analysis include the application of a general purpose modeling language such as CSP to the system security field [12][13], and the use of a graph-theoretic interpretation, called the strand spaces, for protocol verification [14][15].

Trust management is another fundamental area as well as examination of cryptographic protocols to focus on when we study system security in general. Yahalom, Klein and Beth developed a formalism to analyze trust relationships between entities communicating with each other with cryptographic protocols [16]. Because our description method adopts a static trust model to present the structure of the security of systems, trust management is beyond the scope of this paper except considering transferring trust-related elements between entities when entities combine. Adding richer trust management rules to our method is our next challenge to make our method applicable to wider application areas requiring dynamic establishment of trust relationships between entities.

## VI. CONCLUSION

We have developed a description method that diagrams the structure of the security of a system simply and clearly by representing the trust relationship among entities as an abstract object modeled from system components. It includes not only information processing components such as computing devices and users but also communication protocols, with a simple structure consisting of trust elements. The structure of the entity is designed, so that we can express various vulnerabilities of a system uniformly with the possibility of replacing entities without necessarily envisioning any concrete threats against the system. We rely upon properties of security services of cryptographic protocols proven by prior research. We believe this description method can serve as an efficient tool for engineers to design and analyze a system, and a useful communication vehicle among stakeholders of the system as it can describe the security properties of systems clearly. With this method, stakeholders can readily confirm that their security requirements on the system are fulfilled.

## REFERENCES


[1] J¨urjens J. : Towards development of secure systems using UMLsec. Fundamental Approaches to Software Engineering (FASE/ETAPS 2001), LNCS 2029:187–200, 2001.

[2] Mead, N. R., Hough, E. and Stehney, T. : Security Quality Requirements Engineering (SQUARE) Methodology (CMU/SEI-2005-TR009). 2005.

[3] Dahl, E., Hogganvik, I. and Stolen, K. : Structured semantics for the CORAS security risk modelling language. Technical Report. A970, SINTEF ICT, 2007.

[4] ISO/IEC 27002 Information technology. Security techniques. Code of practice for information security management .

[5] IETF, RFC2246, The TLS Protocol, Version 1.0, 1999.

[6] IETF, RFC4120, The Kerberos Network Authentication Service (V5), 2005.

[7] NIST SP800-57, Recommendation for Key Management - Part I: General, 2006.

[8] NIST SP800-63-1, Electronic Authentication Guideline, 2008

[9] Yu, T., Hartman, S. and Raeburn, K. : The perils of unauthenticated encryption: Kerberos Version 4. Proceedings of the Network and Distributed Systems Security Symposium. The Internet Society, 2004.

[10] Dolev, D. and Yao, A. : On the security of public key protocols. IEEE Transactions on Information Theory, 29(2):198-208, 1983.

[11] Burrows, M., Abadi, M. and Needham, R.M. : A Logic of authentication. Proceedings of the Royal Society, Series A, 426(1871):233-271, 1989.

[12] Schneider, S. : Verifying authentication protocols with CSP. Proceedings of the 10th IEEE Computer Security Foundations Workshop, 1997.

[13] Ryan, P. : Mathematical models of computer security. Foundations of Security Analysis and Design, LNCS 2171:1-62, 2001.

[14] Thayer, F. J., Herzog, J. C. and Guttman, J. D. : Strand spaces: Proving security protocol correct. Journal of Computer Security, 7(2/3):191-230, 1999.

[15] Guttman, J. D., Thayer, F. J., Carlson, J. A., Herzog, J. C., Ramsdell, J. D. and Brian, T. S. : Trust management in Strand Spaces: A Rely-Guarantee method. Proceedings of the European Symposium on Programming (ESOP '04), LNCS 2986:325-339, 2004.

[16] Yohalom, R., Klein, B. and Beth, T. : Trust relationships in secure systems – A distributed authentication perspective. Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy:150-164, 1993.