DiSigncryption: An Integration of Agent-based Signature Delegation with Distributed Reputation Management Scheme

O. Bamasak, N. Zhang, D. Edwards
Department of Computer Science
University of Manchester, UK
{obamasak, nzhang, dedwards}@cs.man.ac.uk

Abstract

This paper presents a Distributed Signcryption with Verifiable Partial Signature (DiSigncryption) protocol that allows a mobile agent owner, participating in e-commerce transaction, to securely delegate and distribute his/her signing capability among a set of trusted third party hosts (TTP-hosts) via a mobile agent. The protocol incorporates three schemes: a novel Distributed Reputation Management scheme, a modified version of the Distributed Signcryption method proposed in [11], and the Agent-based Threshold Proxy Signcryption (ATPS) protocol proposed in [1]. The most notable feature of the DiSigncryption protocol is that, in addition to allowing secure distributed proxy signature generation, it enables the agent owner to quantitatively assess the trust and reliability of each of the TTP-hosts that s/he has dealt with. These trust and reliability values are then aggregated into an index to guide the agent owner in making his/her decision as which TTP-hosts should be used in his/her next dealing. The security properties of the proposed protocol are analyzed, and the protocol is compared with the most related work.

Keywords: Proxy signature, security protocols, mobile agent-based e-commerce, reputation management.

1.Introduction

The use of mobile agents to commit to transactions on behalf of a user has recently become a topic of interest. Mobile agents, however, face the problem of execution in a hostile environment where the host executing the agent has access to all the data that an agent carries, e.g. a signature key. Therefore, the problem of enabling an agent to sign a transaction on behalf of its owner, i.e. acting as a proxy signer, has become an attractive research area. Previous research works [6, 7, 10, 12] have proposed some solutions to the above problem. However, these solutions have mainly focused on the protection of the signature key against third party perpetrators, and are weak in tackling threats imposed by the other side of the business deal, i.e. the merchant host. For example, the work by [7, 12] has failed to provide non-repudiation of signature receipt service. The work by [10] does not protect the signature key from being misused by the merchant host. Though Kim [6] has recognized and addressed some of these weaknesses, but the solution proposed is computationally expensive.

However, one problem with the above solutions is that the agent is still given the power to sign a transaction, subject to the requirements/constrains specified earlier by the agent owner. Thus, while the constrains may limit the nature and the value of a transaction, a malicious host may force an agent to commit to a transaction much less satisfactory than could be achieved. To further protect against malicious hosts, an agent owner may wish to employ more than one entity, i.e. trusted third party hosts (TTP-hosts), and have those entities jointly agree on a transaction and sign a relevant document. Hence, an agent owner may delegate N proxy signers (the agent + (N-1) *TTP-hosts*) with criteria that at least T of them performs the required operations correctly. An obvious solution is to employ a threshold proxy signature scheme, which allows the N proxy signers to sign (partially) a document and then, on receipt of a sufficient number of 'correct' partial signatures, the owner's proxy signature can be reconstructed. Vast amount of research has been conducted in the area of threshold signature [4, 8, 9, 15, 16]. However, in these solutions, all the proxy signers, i.e. *TTP-hosts*, are used indiscriminately for each protocol execution without taking into account their past behaviour, i.e. reputation. To further enhance the threshold proxy signature scheme, the agent owner may decide on a group of *TTP-hosts* that have acceptable level of reputation to participate in the transaction.

In this paper, we give a brief description of our novel Distributed Reputation Management scheme suited to the agent-based threshold signature delegation scenario mentioned earlier. This scheme allows an agent owner to assign and update trust and reliability values for each *TTP-host* that the agent owner has dealt with. These values reflect a credit level for the *TTP-*

host over time, and the credit level may increase or decrease depending on the behaviour of the *TTP-host* concerned. This reputation management scheme is then integrated into an extended version of our ATPS protocol [1] and a modified version of the Distributed Signcryption proposed in [11], resulting in a novel Distributed Signcryption with Verifiable Partial Signature (DiSigncryption). It provides a secure and efficient approach to mobile agent-based signature delegation and facilitates proxy signers' (*TTP-hosts*') reputation management.

The remainder of the paper is organized as follows. Section 2 outlines the security requirements for the design of the DiSigncryption protocol. Section 3 introduces the Distributed Trust Management scheme. Section 4 presents the DiSigncryption protocol. In Section 5, the protocol is analyzed against the requirements specified and compared with related work, and finally, our conclusions and future work are given in Section 6.

2. Security Requirements

(S1) *Proxy key confidentiality*: A proxy key delegated to a mobile agent should enjoy confidentiality protection, i.e. to protect it from being disclosed to any single entity, e.g. a merchant host, a *TTP-host*, or any other host or agent.

(S2) *Partial proxy key share Confidentiality:* A proxy key share sh_i should only be revealed to one proxy entity *TTP-host*$_i$.

(S3) *Proxy signature unforgeability:* It should be difficult for an entity other than the agent owner to forge a proxy signature, i.e. to generate a valid proxy signature.

(S4) *Partial proxy signature verifiability:* Partial proxy signatures should be verifiable, i.e. the validity of a partial proxy signature should be verifiable through the use of a *commitment*. This enables the signature verifier to detect and exclude any invalid partial proxy signature during the proxy signature construction process.

(S5) *Non-repudiation of signature origin:* It should be difficult for the original signer (i.e. the agent owner) of a proxy signature to falsely deny that it has delegated the signing power to the agent.

(S6) *Non-repudiation of signature receipt*: It should be difficult for a signature recipient to falsely deny that it has received the proxy signature, if this signature is taken as the proof of a deal agreed between the proxy signer (i.e. the agent) and the recipient.

(S7) *Fairness:* This requirement indicates that, once a deal is agreed, then either the original signer and the signature recipient have both received the proxy signature on the deal, or neither of them has received anything useful.

(S8) *TTP-hosts accountability:* Any misbehavior by a *TTP-host* should be detected and accounted for.

3. Distributed Reputation Management Scheme

In our distributed reputation model, a *TTP-host*'s reputation is measured in terms of a trust level and a reliability level, both of which are aggregated over a specified past period. The trust level reflects the truthfulness of the *TTP-host* in executing a transaction and the reliability level reflects its robustness in providing the TTP service. Both levels are functions of the following parameters: (1) Transaction outcome feedback; (2) Total number of transactions performed; (3) Transaction value; (4) Total number of malicious incidents; and finally (5) Reputation of the source of feedback.

Considering the above parameters, two algorithms are designed to allow an agent owner to distribute a security-sensitive task among a set of N trusted hosts, *TTP-host*$_i$, where $i \in \{1, .., N\}$. The first algorithm, called TTP-hosts Subgroup Selection (TSS) algorithm, allows the agent owner to select a subgroup of Y most trustworthy *TTP-hosts* from N available ones based upon their trust and reliability values. The second algorithm, called Trust and Reliability Updating (TRU) algorithm, allows the agent owner to evaluate and assign trust and reliability values to each *TTP-host* that s/he has employed based upon the feedback received from his/her merchant host. In the following, we give assumptions used for the design of the algorithms. Their detailed description can be found in [2].

- The agent owner maintains a table TA (Trust Assessment) containing trust and reliability values associated with each of the *TTP-hosts* that the agent owner has dealt with in the past period T_h.
- The agent owner also maintain a table MR (Merchant Reputation) containing reputation values associated with each of the merchants that the agent owner has dealt with in the past time period T_m.
- The merchant, once agreed on a deal with the mobile agent, creates a table TM, containing the trust and reliability values for all the participating *TTP-hosts*. The merchant fills table TM with the values correspond to the transaction outcome with each of the participating *TTP-hosts*. The merchant then passes table TM to the agent owner, via the mobile agent for him to update table TA accordingly.
- As mentioned above, we have specified validity periods T_h and T_m for the tables TA and MR, respectively. This can help the agent owner in

maintaining the freshness of the relevant data and reduce memory and computational expenses.

4. DiSigncryption Protocol

In this section, we integrate the Distributed Reputation Management scheme presented in Section 3 to the Agent-based Threshold Proxy Signcryption (ATPS) protocol [1] to derive out novel DiSigncryption protocol. In other words, in the DiSigncryption protocol to be presented shortly, a set of multiple *TTP-hosts* are dynamically selected on per-transaction basis, and these *TTP-hosts* are jointly perform the role of a proxy signer. In the following, we will first summarize the notation, and state the assumptions, used in the protocol design, and then present the protocol formally.

4.1 Notation

The notation to be used throughout the rest of this paper is summarized as follows.

- $H(x)$ is a one-way collision free hash function, e.g., SHA-1.
- $E_k(x)$ and $D_k(x)$ express the encryption and decryption of a data item x using a symmetric key k and a symmetric cryptosystem, e.g., DES or AES.
- $Enc_{pk_i}(x)$ expresses the ciphertext of a data item x encrypted with the public key pk_i using ElGamal public-key cryptosystem.
- $Sig_{sk_i}(x) = (r, s)$ denotes a digital signature on a data item x generated using a private key sk_i of party i and a signature scheme such as DSS.
- sh_i denotes the proxy key share assigned to a party i.
- (c_2, r_2, ps_i) denotes the partial signature generated by party i using a proxy key share sh_i, to be explained in section 4.3.5.
- $Comm_i$ denotes a commitment generated by $TTP\text{-}host_i$ to authenticate its partial signature (c_2, r_2, ps_i).
- $A \xrightarrow{E} B$: m denotes that party A sends party B a message m via an external channel such as a telecommunication network.
- $A \xrightarrow{I} B$: m denotes that party A sends party B a message m internally via an internal message passing mechanism.

4.2 Assumptions

- Every party i ($i \in \{A, B, TTP\text{-}host_k\}$, and $k \in \{1, \ldots, N\}$) has a pair of private and public ElGamal keys, expressed as $sk_i \in_R Z_q^*$ and $pk_i = g^{sk_i}$. The public key pk_i is certified in the form of a digital certificate $Cert(i)$ that is signed by a certification authority (CA) trusted by all parties.
- *TTP-hosts*, in addition to assisting in the proxy signature generation, also assist in proxy signature verification and to store transaction evidences for dispute resolution. It is assumed that *TTP-hosts* may collude with each other, but F out of N *TTP-hosts* are trustworthy.
- B is assumed to provide mechanisms to protect the mobile agents it hosts from being eavesdropped on their contents by other agents hosted also by B. B can use existing solutions, e.g. tamper-resistant hardware [14] and time limited blackbox security [5], to provide such mechanisms.

4.3 Protocol Description

The DiSigncryption protocol has beautifully integrated the Distributed Reputation Management scheme presented in Section 3 and the cryptographic primitives presented in Section 4.3 to achieve distributed agent-based proxy signature delegation/generation. Here, in the following, the protocol is described as a seven-step procedure.

Step 1 – Execution initialization: During this stage, the agent owner specifies the shopping requirements and generates the parameters needed for the proxy signature delegation. In detail, A performs the following setup operations prior to the protocol execution.

1. A executes the TSS algorithm to select a subset of *TTP-hosts* from table TA, i.e. the *AS* list containing Y members, which would satisfy the risk threshold specified.
2. A generates a group public key G for the Y participating *TTP-hosts* [11] as follows. A first generates Y secrets $x_i \in Z_p$, and send each x_i securely to the corresponding $TTP\text{-}host_i$. Only A and $TTP\text{-}host_i$ have the knowledge about the secret x_i. A then construct a polynomial function of order Y as follows:

$$f(x) = \prod_{i=1}^{Y}(x - x_i) \equiv \sum_{i=0}^{Y} a_i x^i \pmod q, \tag{2}$$

where the set $\{a_i\}$, $i \in \{0, .., Y\}$ are the coefficients .It is worth noting that $\sum_{i=0}^{Y} a_i x_j^i = 0$. Having obtained the set $\{a_i\}$, A constructs the corresponding exponential values, i.e. the group public key $G = \{g^{a_0}, g^{a_1}, \ldots, g^{a_Y}\} \equiv \{g_0, g_1, \ldots, g_Y\}$. All elements are computed under modulo p. Note that $\prod_{i=0}^{Y} g_i^{x_j^i} = 1$.

3. A prepares a document M_A that specifies the purchase or signature generation requirements, e.g. description of goods to be purchased or the contract

to be signed. M_A is signed with A's private key sk_A using DSS.

4. A generates a proxy key pr_A from its private key sk_A as described in [1]. A then constructs a message M containing the signed M_A along with items for proxy signature verification, i.e. the values to be used by the *TTP-hosts* for the verification of the proxy signature (α and w), the transaction identifier I_t, and the proxy key's validity period V, i.e. $M = (Sig_{sk_A}(M_A), \alpha, w, I_t, V)$.
5. A then generates $(Y+1)$ shares sh_i, $i \in \{MA, 1, \ldots, Y\}$ as described in [1]. A also hashes the values of these shares, i.e. $SH = \{H(sh_1), H(sh_2), .., H(sh_Y)\}$that will be used by the *TTP-hosts* to verify the integrity of the received shares.
6. A signcrypts (M, SH) as follows [3]. A first blinds the shares sh_i, $i \in \{1, .., Y\}$ by computing:

$$u_i = (sh_i \times x_i) \qquad (6)$$

A then signcrypts (M, SH) by first choosing a random number $x \in Z_q$, which is kept secret by A, and then computes the following for $i \in \{1, .., Y\}$:

$$k = g^x \bmod p \qquad (7)$$

$$c_1 = E_k(M, SH) \qquad (8)$$

$$r_1 = H(c_1, k) \qquad (9)$$

$$s_i = x / ((kr_1 + sk_A) \times u_i) \bmod q \qquad (10)$$

(c_1, r_1, s_i) represents the signcryption of (M, SH) using the private key sk_A and the blinded share u_i. A then send the signcrypted message (c_1, r_1, s_i) together with the group public key G and u_i to $TTP\text{-}host_i$, $i \in \{1, .., Y\}$.

7. A loads MA with the following message:

T1. $A \xrightarrow{I} MA: ((MAitem, Sig_{sk_A}(MAitem))$

Where $MAitem = (ID_A, sh_{MA}, c_1, r_1, G, S, U)$, $S = \{s_1, s_2, .., s_Y\}$,and $U = \{u_1, u_2, .., u_Y\}$.

7. A then dispatches MA in to the network to search at various merchant hosts for a suitable offer.

Step 2 – Offer searching and proxy key share distribution. If MA finds a suitable offer, say M_B,, at a merchant host B, then B will provide MA with an execution environment so that MA will run locally to execute the rest of the protocol. Residing at B, MA generates a random number *rand* and sends the message $(c_1, r_1, s_i, rand, G, u_i, M_B)$ to each of the Y *TTP-host* via a secure channel, e.g. SSL [13], and sends U to B internally for it to perform partial signature verification.

T2.1. $MA \xrightarrow{E} TTP\text{-}host_i$: $(c_1, r_1, s_i, rand, G, u_i, ID_B, M_B)$, $i \in \{1, .., Y\}$

T2.2. $MA \xrightarrow{I} B$: U

Step 3 – Partial proxy signature generation and delivery. Each $TTP\text{-}host_i$, once received message T2.1, performs the following verification:

<u>*Verification TTP-host-1:*</u>

Check the correctness of A's signature on the signcryption (c_1, r_1, s_i) received in T2.1. This is done by recovering the key k using equations (11) and (12), and then checking if $H(c_1, k) = r_1$.

If the verification fails, $TTP\text{-}host_i$ sends an error message to *MA* asking it to resend the message T2.1. If the verification fails for the second time, $TTP\text{-}host_i$ will send an error message to both *MA* and *B* and terminate the protocol execution. Otherwise, if the verification is positive, $TTP\text{-}host_i$ proceeds by performing the following operations.

- Decrypts c_1 to reveal (M, SH):

$$(M) = D_k(c_1) \qquad (13)$$

- Unblinds the proxy key share sh_i :

$$sh_i = u_i / x_i \qquad (14)$$

$TTP\text{-}host_i$ then performs the following verification:

<u>*Verification TTP-host-2:*</u>

(a) Check the correctness of B's signature on M_B using the DSS signature verification algorithm].

(b) Check if the conditions specified in M_B matches with that in M_A contained in M.

(c) Check if the values, I_t α, and w in M are fresh (i.e. they do not already exist in $TTP\text{-}host_i$'s database) and the time of the arrival of T2.1 is within the validity period V specified in M.

(d) Verifies the integrity of the recovered proxy key share sh_i, i.e. check that the hash of the recovered share $(H(sh_i))$ exists in SH received in T2.1.

If any of the steps in *Verification TTP-host-2* is negative, $TTP\text{-}host_i$ should send an error message to B and terminate the protocol execution. Otherwise, if the verifications is all positive, $TTP\text{-}host_i$ will compute a partial signature by performing the following calculation:

$$k = H(pk_B^{rand} \bmod p) \qquad (15)$$

$$y_1 = g^{rand} \; mod \; p \qquad (16)$$

$$c_2 = E_k(Doc) \qquad (17)$$

$$r_2 = H(y_1, c_2) \qquad (18)$$

$$l_i = sh_i \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i \quad ID_j} mod \; q, \qquad (19)$$

$$ps_i = rand / (r_2 + Yl_i) \bmod q \qquad (20)$$

$$Comm_i = g^{x_i^{\,1} \times \prod_{j=1, j \neq i}^{Y} \frac{IDj}{ID_i \quad ID_j}} mod \; p \qquad (21)$$

Here, the partial signature on *Doc* is (c_2, r_2, ps_i), where $Doc = (ID_A, ID_B, M_A, M_B, I_t)$. $TTP\text{-}host_i$ also computes a commitment $Comm_i$ that B will use to verify the partial signature using equation (21). The partial signature (c_2, r_2, ps_i) and the commitment $Comm_i$ are signed with $TTP\text{-}host_i$'s private key $sk_{TTP-host_i}$ and sent to B. Similarly, MA computes the partial signature (c_2, r_2, ps_{MA}) on *Doc* with its share sh_{MA} and sends it internally to B. That is,

T3.1 $TTP\text{-}host_i \xrightarrow{E} B$:

$$((c_2, r_2, ps_i, Comm_i), Sig_{sk_{TTP-host_i}}(c_2, r_2, ps_i, Comm_i))$$

or Error message

T3.1 will be executed by all $TTP\text{-}host_i$, $i \in \{1,.., Y\}$.

T3.2. $MA \xrightarrow{I} B$: (c_2, r_2, ps_{MA})

Step 4 – Partial proxy signature verification and complete proxy signature construction: Upon the receipt of message T3.1, B performs Verification B-1 as follows.

<u>Verification B-1:</u>

(a) Check the correctness of TTP-host$_i$'s signature on T3.1 using the DSS signature verification method.

(b) Check the validity of the received partial proxy signatures as follows:

$$V = Comm_i^{Y \times u_i} = \left(g^{x_i^{-1} \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j}}\right)^{Y \times sh_i \times x_i} \quad (22)$$

$$= g^{Y \times sh_i \times x_i \times x_i^{-1} \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j}}$$

$$= g^{Y \times sh_i \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j}} \ mod\ p$$

B then computes:

$$T = (V \times g^{r_2})^{ps_i} = \left(g^{Y \times sh_i \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j} + r_2}\right)^{ps_i} \quad (23)$$

$$= g^{\left(Y \times sh_i \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j} + r_2\right) \times \left(\frac{rand}{r_2 + Y \times sh_i \times \prod_{j=1, j \neq i}^{Y} \frac{ID_j}{ID_i - ID_j}}\right)}$$

$$= g^{rand} \bmod p$$

Finally, B compares H(T mod p, c_2) with r_2, i.e. to confirm if (c_2, r_2, ps_i) is indeed generated with the right key share sh_i.

If the above verification is successful for a particular partial signature, then the generating host of this partial signature will be confirmed as trustworthy for this transaction and will get the credit accordingly. Otherwise, if any of the above verifications fail, B will send an error message to the *TTP-host* concerned and request for a retransmission. If repeated negative verification occurs, then this *TTP-host* will be branded as "dishonest" and get penalty accordingly. That is, B will fill the table TM with the corresponding value for the trust and reliability attributes for each $TTP\text{-}host_i$. For *TTP-host*(s) with positive verification results, the value of *Trust* will be set to 'Yes'. Otherwise, the *TTP-host*(s) with negative verification results will get a 'No' for their/its *Trust* value. In addition, if no message is received from a particular *TTP-host* then its *Trust* value will be set to 'Unknown'. The reliability value will be determined according to whether or not B has actually received (c_2, r_2, ps_i) at all. If B has received it, then the *Reliability* value will be set to 'Yes', otherwise, it will be set to 'No'.

In the case that B has received at least F out of Y valid partial signatures from Y *TTP-hosts*, B will proceed to construct the complete proxy signature, (c_2, r_2, S_{pr_A}), as explained in [1]. As B does not posses the items needed for the verification of the proxy signature, B has to forward the newly constructed proxy signature (c_2, r_2, S_{pr_A}) to the *TTP-hosts,* each of which will verify the signature.

T4. $B \xrightarrow{E} TTP\text{-}host_i$:

$$(c_2, r_2, S_{pr_A}, Sig_{sk_B}(c_2, r_2, S_{pr_A}))$$

Step 5 – Proxy signature verification and token generation. In this stage, *TTP-host(s)* perform proxy signature verification upon B's request, and generate a verification token (*VT*) accordingly [1], which proves that the deal has been signed and B has received MA's proxy signature. Upon recipient of T4, $TTP\text{-}host_i$ performs the following verification.

<u>Verification TTP-host-3:</u>

(a) Check the correctness of B's signature on T4 using the DSS signature verification method.

(b) Check the correctness of the proxy signature (c_2, r_2, S_{pr_A}) using the Proxy Signature Verification method

If any of the above verification fails, $TTP\text{-}host_i$ will send an error message to B to request for retransmission. If repeated retransmissions still result in a negative verification outcome, $TTP\text{-}host_i$ will send an error message indicating that the proxy signature is invalid and terminates the protocol run. In this case, B will forward this error message to MA, which will be delivered in turn to A. If the verification is positive, which indicates both the validity of the signature (c_2, r_2, S_{pr_A}) and the authenticity of A's delegation (by using A's public key pk_A in the verification of the proxy signature), $TTP\text{-}host_i$ will generate and send to B, in Transaction T5, a signed and time-stamped verification token *VT*. This *VT* can be used to prevent B from false denial of signature receipt thus supporting non-repudiation of the receipt of A's signature by B.

T5. $TTP\text{-}host_i \xrightarrow{E} B$: *VT* or *Error message*

Where, *VT* =

$$(c_2, r_2, S_{pr_A}, Sig_{sk_B}(c_2, r_2, S_{pr_A}), T_{TTP-host_i}, ID_A, ID_B)$$

$, Sig_{sk_{TTP\ host_i}}(c_2, r_2, S_{pr_A}, Sig_{sk_B}(c_2, r_2, S_{pr_A}),$

$T_{TTP\ host_i}, ID_A, ID_B)$

and $T_{TTP-host_i}$ is a timestamp generated by *TTP-host$_i$* to indicate the time when the *VT* is generated.

Step 6 – Signing and returning the token to *MA*. Merchant *B*, once obtained the verification token *VT*, confirms *TTP-host$_i$*'s signature on *VT*. If the verification is positive, *B* submits the *VT* and table *TM* to *MA* that will return back to its owner *A*. If *B* receives an error message in T5, it forwards this error message to *MA*.

$$\textbf{T6}. B \xrightarrow{I} MA:$$

$$(VT, TM, Enc_{pk_A}(rand), Sig_B(VT, TM, Enc_{pk_A}(rand))$$

or Error message

Step 7 – Protocol execution completion and final verification. In this stage, *MA* returns home and passes the signed deal to its owner *A*. If *A* receives the signed *VT* in T6 rather than the error message, *A* performs the following verification to confirm the signed deal:

Verification A:

a) Check the correctness of B's signature on T6 using the DSS signature verification method.

b) Check the correctness of TTP-host$_i$'s signature on VT using the DSS signature verification method.

c) Check the correctness of the proxy signature S_{pr_A} in VT using the Proxy Signature Verification method.

Verification *A* being positive means that *A* has conducted a valid deal with *B,* and both *A* and *B* have received a valid document *Doc* signed by both parties. *A,* in this case, retrieves the plaintext *Doc* from the signcryption (c_2, r_2, S_{pr_A}) in *VT* by performing the following tasks:

1. *A* decrypts the ciphertext $Enc_{pk_A}(rand)$ with the private key sk_A to reveal *rand*.
2. *A* then uses *rand* together with *B*'s public key pk_B to generate *k* using equation (15).
3. *A* finally uses *k* to decrypt the ciphertext c_2 and reveal the plaintext *Doc*.

If the outcome of Verification *A* is positive, *A* stores the *VT* as it has both his and *B*'s signatures on *Doc*, approved by a *TTP-host*. *A* also updates the table *TA* according to the data in table *TM* by executing TRU algorithm. If Verification *A* is negative, which means that *A* has failed to obtain a correct *VT*, *A* initiates a recovery protocol with a *TTP-host* to recover *VT*. A detailed description of the recovery protocol can be found in [1].

5. Analysis of the DiSigncryption protocol

5.1 Comparison with related work

To highlight the merits of our DiSigncryption protocol, the efficiency, reliability, robustness, and accountability of our protocol is compared with that of related distributed (multiple) *TTP-hosts* based protocols. The recently proposed protocols by Hsu et al. in [15], referred to as Hsu's protocol, and by Tzeng at la. In [16], referred to as Tzeng's protocol, are chosen as samples of the distributed *TTP-host* based approach. The reason for choosing these protocols is that they are all designed to perform the same task as ours. That is, an original signer delegates its signing power to proxy signer(s). Any *T* or more out of the *N* proxy signers can cooperatively reconstruct and verify the proxy signature on the message, but (*T*-1) or fewer proxy signers cannot.

Table 1 shows the comparison between the DiSigncryption protocol and the three related works mentioned above in terms of communication overheads, measured in terms of the number of messages exchanged among the protocol entities and their sizes. The following assumptions have been used in calculating a message size:

- AES algorithm is used for symmetric encryption, therefore, the size of a ciphertext is in multiples of 128 bits.
- The key used for AES is 192 bits long.
- SHA-1 algorithm is used for one-way hashing, therefore, the output of the hashing process $|H()|$ is 160 bits.
- The prime $|p|$ = 1024 bits, $|q|$ = 160 bits
- The identifiers used in the protocol, e.g. ID_A, are 32 bits in size.
- The sizes of the agent owner's requirements M_A and the remote host's offer M_B are 512 bytes each. Therefore the size of $Doc = (ID_A, ID_B, M_A, M_B, I_t)$ is 1024 bytes.

Table 2 shows the computational overhead measured in terms of the total number of multiplication and exponentiation operations performed at each protocol step. The quantitative results in both tables are materialized by evaluating them with the total number of participating *TTP-hosts* (*N*) = 10, the minimum threshold of active *TTP-hosts* (*T*) = 5, and the number of *TTP-hosts* in the *AS* list (*Y*) = 5.

Table 1. Communication overhead.

	Number of messages	*Size of messages (bytes)*
Hsu	280	75440
Tzeng	165	18800
DiSigncryption	28	32269

Table 2. Computation overhead.

	Mult	Exp
Hsu	721	1327
Tzeng	911	981
DiSigncryption	227	200

In comparison with Hsu's and Tzeng's protocols, our DiSigncryption protocol achieves 90% and 77% reduction in the number of messages exchanged, respectively. This is an advantageous feature when applied in mobile network environment, which is characterized by low and/or expensive bandwidth and higher error rate. In addition, the total size of our protocol messages is approximately 57% less than that of Hsu's scheme. However, the total size of our protocol messages is 41% higher than that of Tzeng's protocol due to the fact that our protocol provides four extra security services, non-repudiation of signature receipt, fairness for both the agent owner and the merchant, confidentiality of the signed message, and the accountability of *TTP-hosts* service, which neither Hsu's nor Tzeng's protocol provides. These features/services are necessary for e-/m-commerce applications. Regarding the computational costs, the DiSigncryption protocol enjoys a saving of approximately 84% and 79% in the number of performed exponentiation operations, and 68% and 75% in the number of performed multiplication operations, comparing with Hsu's and Tzeng's protocols, respectively.

5.2 Security Analysis

In this section, we analyze the security properties of the DiSigncryption protocol showing that it satisfies all the security requirements stated in Section 3.

- Proxy key confidentiality: It is difficult to compromise the proxy key pr_A due to the following reasons: (1) the proxy key is distributed in (Y+1) shares, and (2) each share is blinded with a secret x_i that is known only to A and the $TTP\text{-}host_i$. In order to compromise the proxy key, one has to intercept and brute force attack at least F blinded shares u_i, of which the security relies on the difficulty of factoring large primes unless F or more *TTP-hosts* collude together.
- Proxy key shares confidentiality: Each share sh_i is blinded by the secret x_i, and both A and $TTP\text{-}host_i$ are the only parties that have knowledge of x_i, so only A and $TTP\text{-}host_i$ have knowledge of this share. The security of u_i depends on the difficulty of factoring large primes, i.e. factoring u_i to get sh_i and x_i.
- Proxy signature unforgeability: Since the proxy key pr_A is derived from A's private key sk_A, it would be difficult for another party to forge the proxy key without knowledge of A's private key. In addition, as multiple *TTP-hosts* are involved in the proxy signature generation process, each *TTP-host* has only a share of the proxy key, and this share can only be used to generate a partial signature, it would be difficult for a single *TTP-host* to forge a valid proxy signature on *Doc* without colluding with others.
- Partial proxy signature verifiability: B is able to verify the validity of each partial proxy signature (c_2, r_2, ps_i) using the commitment $Comm_i$ generated by $TTP\text{-}host_i$ and the corresponding u_i received from A through the mobile agent *MA*. It is worth noting that B is able to verify (c_2, r_2, ps_i) without accessing plaintext sh_i. This feature supports the confidentiality of the proxy key shares and hence protects the proxy key from being disclosed to any non-holding parties including B.
- Non-repudiation of signature origin: The verification *Verification TTP-host-1* performed by the *TTP-hosts* ensures that the proxy signature S_{pr_A} on *Doc* is generated by using a proxy key that is generated from A's private key, and that the proxy signature verification requires the use of A's public key. Therefore, A cannot deny the fact that he has generated the proxy key.
- Non-repudiation of signature receipt: This requirement is achieved through the use of a verification token *VT* signed by the *TTP-hosts* and sent to A through *MA* in T6. As B cannot verify the proxy signature, B has to send a signature verification request to the $TTP\text{-}host_i$, which proves that B has actually received A's proxy signature on *Doc* if the verification is positive and the token *VT*, signed by the $TTP\text{-}host_i$ is produced. Therefore, B cannot deny later that it has received A's proxy signature on *Doc*.
- Fairness: Our protocol achieves the fairness requirement. This is illustrated by the following scenario. B, after receiving *VT* from the *TTP-hosts* in T5, may attempt to cheat A by sending an incorrect *VT* (for a deal between B and another party Z, for example) or a bogus message to *MA*. In other words, B may attempt to get A's signature (verified proxy signature) on *Doc* but refuse to hand out its own one. A can discover this attempt when performing *Verification A* and consequently initiate the recovery process with a *TTP-host* to retrieve *VT*.
- Confidentiality of the document to be signed: The confidentiality of document *Doc* while it is being transferred between protocol's parties (A, *MA*, B, and *TTP-hosts*) is achieved by using the following measures.

  (1) The communication channels between B and *TTP-hosts* are secured using SSL that protects the confidentiality of the messages transmitted through them.

(2) The items sent through the channels between A and B, i.e. T1 and T6, are secured using signcryption.
(3) The items sent through the channels between *MA* and B, i.e. T2.2, T3.2, and T6, are either contain no useful information regarding the contents of *Doc* or secured using signcryption.

• *TTP-host* accountability: This security requirement is addressed through the use of Verification B-1 and our proposed award/penalty mechanism. The outcome of Verification B-1 performed by B in the verification of the partial proxy signature (c_2, r_2, ps_i) received from $TTP\text{-}host_i$ will indicate if $TTP\text{-}host_i$ has followed the protocol execution correctly and credit/penalize it accordingly.

6. Conclusion

This paper has addressed the distributed reputation management issue by critically analyzing related works and highlighting their shortcomings. We then presented a novel Distributed Reputation Management scheme, which enables a party A, i.e. a customer, to distribute a security sensitive task among several *TTP-hosts*. This is achieved by first choosing a subset of *TTP-hosts* with the highest trust and reliability levels. The scheme then credits/penalizes each *TTP-host* according to a feedback received by A from party B, e.g. a merchant. The paper then presented a novel DiSigncryption protocol, which integrates into it the Distributed Reputation Management scheme, a modified version of the Distributed Signcryption proposed in [11] and an extended version of the ATPS protocol proposed in [1]. The new protocol has the following features. Firstly, it enables the agent owner to delegate signing power to its mostly trusted subset of *TTP-hosts* depending on the risk level valued upon transaction values. Secondly, it enables the signature combiner, i.e. the merchant, to verify each partial signature received from a *TTP-host* without access to the plaintext proxy key share, and verification outcome is used to rate the *TTP-host*'s honesty and credit/penalize it accordingly. Thirdly, it has an embedded algorithm to allow the agent owner to update the trust and reliability values for each *TTP-host*. The protocol analysis shows that, in addition to fulfilling the security requirements specified in section 3, it is more flexible and robust in comparison with the related work. Our protocol can be applied to many applications, e.g. e-/m-commerce, grid computing, and ubiquitous computing due to the properties of mobile agents.

The future work will be the formal verification of the security properties of the proposed protocol.

References

[1] O. Bamasak and N. Zhang, "A Secure Proxy Signature Protocol for Agent-Based M-Commerce Applications", In *Proceedings of the 9TH IEEE Symposium on Computer and Communications*, 2004, pp. 399-406.

[2] O. Bamasak and N. Zhang, "A Distributed Reputation Management Scheme for Mobile Agent-based E-commerce Applications", to appear in *Proceedings of IEEE Conference on e-Commerce, e-Technology and e-Service (EEE-05)*, 2005

[3] C. Gamage, J. Leiwo, and Y. Zheng, "Encrypted message authentication by firewalls", In *Proceedings of PKC'99*, LNCS 1560, Springer-Verlag, 1999, pp. 69-81.

[4] R. Gennaro, J. Stanislaw, H. Krawczyk, and T. Rabin, "Robust Threshold DSS Signatures" In *Advances in Cryptology*, LNCS 1070, Springer-Verlag, 1996, pp.354-371.

[5] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from malicious Hosts", In *Mobile Agents and Security*, LNCS 1419, Springer-Verlag, 1998, pp. 92-113.

[6] H. Kim, J. Baek, B. Lee, and K. Kim, "Secret Computation with Secrets for Mobile Agent using One-Time Proxy Signature", In *Proceedings of the Symposium on Cryptography and Information Security*, 2001, pp. 845-850.

[7] P. Kotzanikolaou, M. Burmester, and V. Chrissikopoulos., "Secure Transactions with Mobile Agents in Hostile Environments", In *proceedings of the Fifth Australian Conference on Information Security and Privacy*, LNCS 1841, Springer-Verlag, 2000, pp. 289-297.

[8] S. Langford, "Threshold DSS Signatures without a trusted party", In *Advances in Cryptology, Proceedings of Crypto' 95*, LNCS 963, Springer-Verlag, 1995, pp. 397-409.

[9] N. Lee, "Threshold signature scheme with multiple signing policies", In IEE Proceedings – Computers and Digital Techniques, 148 (2), pp. 95-99.

[10] B. Lee, H. Kim, and K. Kim, "Strong Proxy Signature and its Applications", In *proceedings of The 2001 Symposium on Cryptography and Information Security*, 2001, pp. 603-608

[11] Y. Mu, and V. Varadharajan, "Distributed Signcryption", In *Proceedings of INDOCRYPT 2000*, LNCS 1977, Springer Verlag, 2000, pp. 155-164.

[12] T. Sander, and C. Tschudin, "Protecting Mobile Agents against Malicious Hosts." In *Mobile Agents and Security,* LNCS 1419. Springer-Verlag, 1998, pp. 44-60.

[13] Stallings W., "Network Security Essentials: Applications and Standards." Prentice Hall, 2000.

[14] U. Wilhelm, "Cryptographically Protected Objects" *Technical report*, Ecole Polytechnique Federale de Lausanne, Switzerland, 1997.

[15] C. Hsu, T. Wu, and T. Wu, "Improvement of threshold proxy signature scheme", In *Applied Mathematics and Computation*, 136, Elsevier Science, 2003, pp. 315 – 321.

[16] S. Tzeng, M. Hwang, and C. Yang, "An Improvement of nonrepudiable threshold proxy signature scheme with known signers". In *Computer & Security*, 23, Eelsevier Science, 2004, pp. 174 – 178.