A New Signature Scheme: Joint-Signature

Li-Sha HE, Ning ZHANG
Department of Computer Science
The University of Manchester
Manchester, UK
0044-161-2756270
{hel, nzhang}@cs.man.ac.uk

ABSTRACT

A number of asymmetrical payment methods have been developed to enable mobile users to buy goods online by charging them to their mobile phone bills by their network operators. It has been recognized that these methods must be used in conjunction with the security services of authentication and non-repudiation of the origin of the request(s) sent from a mobile user so as to prevent fraudulent actions by the network operator or any other entities. This paper presents a novel joint-signature scheme that can be used as one of the security primitives to address the above security services. The scheme enables a mobile user to securely and efficiently instruct his/er network operator for m-payment related actions. It is based on the use of the one-way hash function and traditional digital signature method, but in a collaborative manner with the network operator. The joint-signature scheme achieves the same security services as those by a traditional digital signature scheme, i.e. message origin authentication, message integrity and non-repudiation of origin, but offers lower computational cost for the mobile user. In addition, it imposes lower communication cost in comparison with proxy/server-aided signature schemes.

Categories and Subject Descriptors

C.2.8 [**Communication/Networking and Information Technology**]: Mobile Computing – *algorithm/protocol design and analysis, support services.*

Keywords

Security protocol, Digital signature, M-Commerce, M-Payment.

1. INTRODUCTION

An electronic alternative of a hand-written signature is a digital signature. The advent of digital signatures is changing the way by which people identify themselves, sign legal agreements and conduct business. It enlarges the universe of online commerce of goods and services to high value transactions such as home purchases, online stock trading and government procurement. Today, digital signatures are regarded as one of the most important technologies of the Internet Age. However, existing digital signature schemes are costly for the emerging mobile applications, e.g. mobile payments, run on resource-limited mobile devices. Among those well-known cryptographic operations, digital signature generation is the most time- and resource-consuming operation to be performed by mobile devices [3].

Over the past few years, a number of asymmetrical payment methods and systems have been developed to enable mobile users to buy goods online by charging them through their mobile phone bills by network operators. A notable advantage offered by this approach is that it does not require mobile users to store electronic money (e-money) on their mobile devices, which eliminates a range of security problems related to the storage, transmission and access to the e-money. Nor does it require the use and secure transmission of credit or debit card numbers over the air interface. In other words, this approach reduces the use of mobile phone resources and the security risks. However, a major problem with the approach is that the network operator may abuse the trusts. In addition, there may be disputes between a mobile user and his/er network operator such as whether an m-payment instruction has indeed be given by the user or by a fraudulent employee at the operator. Therefore this approach must be equipped with a strong level of accountability service that holds all the parties (i.e. the user, the network operator or the payee) accountable for their actions/instructions.

The example given next illustrates the need for the accountability security service. Suppose a mobile customer, Alice, wants to buy a train ticket from Virgin Trains Online (an online shop run by a train company to sell train tickets). For this purchase, Alice would like to instruct her network provider, British Telecom (BT) UK to pay for her ticket and she will be charged for the service by BT as part of the payment on the use of her phone. Using a traditional digital signature method, Alice signs the ticket-purchasing request with her private key using her mobile phone before sending the request to BT. The digital signature provides the functionality of identifying the originator of the request (i.e. Alice), detecting any unauthorized alteration to the contents of the request (e.g. the cost of the ticket) and protecting against repudiation of the origin of the request (e.g. Alice may deny sending the request). Upon receipt of the signed request, Virgin Trains Online verifies the signature and issues the ticket to Alice. In addition, BT verifies the signature, pays to Virgin Trains Online on Alice behalf and bills Alice via her mobile phone. However, this idea is not desirable since the digital signature generation is computationally expensive for a mobile device, which has considerably less computing resource than a desktop.

The signature scheme presented in this paper is a more efficient alternative to the traditional signatures in that it costs less at one end. It is based upon the following hypothesis. If a third party with sufficient computational and communication resources (e.g. a host managed by BT) generates/signs a digital signature containing a secret that is shared only between Alice (i.e. the signature originator) and Virgin Trains Online (i.e. the signature verifier), then Virgin Trains Online can treat the digital signature on the secret jointly as a joint-signature originated from Alice but signed by BT. If, firstly, BT can ensure that the secret received is indeed from Alice without knowing it, and, secondly Virgin Trains Online can verify the correctness of both the signature signed by BT as well as the secret contributed by Alice, then Alice is able to instruct BT for m-payment related actions securely with reduced computational cost. This novel joint signature scheme makes the signature generation easy to implement in a mobile device in comparison with other signature generation methods [1, 9, 16].

The remainder of this paper is structured as follows. Next section discusses the related work and the motivation for the design of the joint-signature scheme. Section 3 outlines the notation used in the design of the joint-signature scheme and presents the details of the scheme. Section 4 gives the security and performance analyses of the scheme and compares it with related work. Section 5 outlines the conclusions of our work.

2. RELATED WORK

The concept of *traditional digital signatures* was first invented by Diffie and Hellman [4]. Digital signature provides origin authenticity and detects any unauthorized change to the signed message and therefore serves as non-repudiation evidence. A major problem with *traditional digital signatures* lies in the cost of signature generation. A number of schemes have been proposed to alleviate the problem by designing efficient mathematical algorithms [6, 2]. Many other proposals have been put forward, e.g. allowing a third entity to sign messages on the original signer's behalf. Next we present a survey of these works.

A proxy signature scheme allows a proxy signer to generate a digital signature on behalf of its original signer. The concept was first introduced by Mambo, Usuda and Okamoto in 1996 [10]. Proxy signatures can be classified into three categories based on the types of delegation, i.e. full delegation, partial delegation or delegation by warrant. In full delegation, the proxy signer signs messages with the key of the original signer. In partial delegation, the original signer creates a new proxy key from the original signature key and sends it to the proxy signer. The proxy signer then uses the proxy key to sign messages for the original signer. However, these two schemes make it possible for the original signer to generate a proxy signature as the original signer can make use of its private key or derive the proxy signature key from his original key [16]. Therefore non-repudiation is not provided to identify exactly who (the original signer or the proxy signer) has actually signed the proxy signature. In delegation by warrant [14], a warrant is used together with the proxy signature to certify that the signing power has delegated to the proxy signer. This latest scheme has higher processing overhead as it requires the original signer to sign the certificate with its private key. In order to solve the problem of non-repudiation in partial delegation, *threshold proxy signatures* were proposed [8]. A (t, n) threshold proxy signature scheme allows t or more proxy signers from a designated group of n proxy signers to sign a message on behalf of an original signer. Sun *et al.* [16] pointed out that the threshold proxy signature schemes suffer from an attack in which the original signer is able to execute the (t, n) threshold proxy signature scheme to generate a valid proxy signature by itself. As a result, this scheme does not actually solve the problem of non-repudiation. *Undetachable signatures* proposed by Kotzanikolaou *et al.* [9] are designed for signature delegation in mobile agent paradigm. The scheme allows an agent to sign a digital signature with the original signer's private key without exposing this key to the agent. However, the agent may deny signing the signature as the identity of the agent is not cryptographically verifiable from the resulting signature.

Server-aided signature proposed by Matsumoto *et al.* [12] has drawn a lot of interests among international researchers [5, 1]. *Server-supported signature*, called S^3, presented by Asokan *et al.* [1] is a server-aided protocol based on hash functions and traditional digital signature methods. It introduces a new entity, i.e. the signature server, in the scheme and requires round-trip communication between the original signer and the signature server. In addition, S^3 requires the signature server to verify the signature on the received public key to ensure that the received public key is indeed from the original signer, i.e. the authenticity of the received public key. These introduce communication and computation overheads to the signature generation, and thus increase the overall time taken to generate a signature.

Here in this paper, we propose an alternative solution to those mentioned above. In the mobile application context, such as cellular telephony and wireless web access, mobile devices interact with a fixed and wired infrastructure, which allows the mobile devices to access the Internet and its resources via wireless access networks. It is thus logical for a mobile device to harvest the ample computing and bandwidth resources within the wired network backbone and allow a wired entity, such as the wireless controller or operator, to sign messages, or perform other actions, on behalf of the mobile user. We aim at finding such a solution for signature signing with security and accountability properties.

3. JOINT-SIGNATURE SCHEME

3.1 Notation

The notation to be used for the presentation of the joint-signature is given as follows:

- $x \| y$ denotes the concatenation of data items x and y.
- $h(x)$ is a strong collision resistance hash function, such as SHA-1[13].
- $A \rightarrow B$: m denotes that party A sends a message m to party B.

3.2 Protocol Overview

The overall operation of the joint-signature scheme is illustrated in Figure 1. In the scenario, the originator is a Mobile Station (MS) which generates the message. The signer is a server run by the network operator in its Home Environment (HE). The server may also be in a Serving Network. The verifier is a Service

Provider (SP) which provides a suite of e-services related to m-commerce.

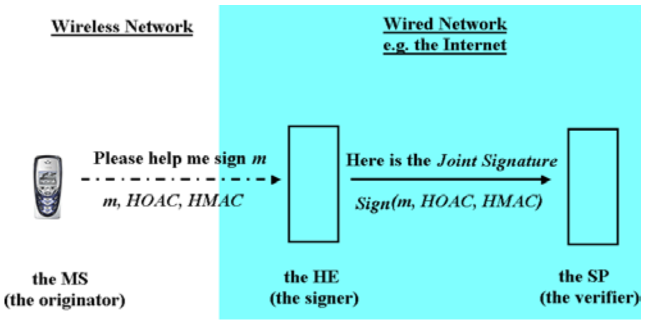

Figure 1. Our joint-signature system overview

Suppose the MS intends to send a message m to the SP and to produce a joint-signature on m with the help of the HE. The joint-signature is signed by the HE using its private key on a *Hash Origin Authentication Code* (*HOAC*), a *Hash Message Authentication Code* (*HMAC*), and the message m, which are generated by the MS. The *HOAC* indicates to the SP that the originator of m is the MS and the *HMAC* indicates to the HE that the *HOAC* is from the MS, as to be detailed in Section 3.3. The *HOAC* and the *HMAC* are crucial for the joint-signature concept. In Section 3.3 next, we explain the generation and verification of the *HOAC*, the *HMAC* and the joint-signature using the widely used RSA algorithm [15].

3.3 Formal Presentation

The joint-signature scheme consists of three processes: initialization, joint-signature generation and joint-signature verification.

3.3.1 Initialization

The purposes of the initialization process are two folds. First, a secret is generated and shared by the MS and the SP. Secondly, a hash of a secret, which is shared between the MS and the HE, is sent to the SP by the HE. One way of doing this is by using the NAETEA protocol proposed by He and Zhang [7]. After an execution of the NAETEA protocol, the MS and the SP have a shared secret, $K_{MS\text{-}SP,\ MS}$ ($=K_{MS\text{-}SP,\ SP}$), and the SP has the hash of $K_{MS\text{-}HE,\ MS}$, i.e. $h(K_{MS\text{-}HE,\ HE})_{SP}$ ($=h(K_{MS\text{-}HE,\ MS})$). For clarity, the definitions of all the items used in the joint-signature are summarized in Table 1.

3.3.2 Joint-Signature Generation

Step 1. The MS produces a *HOAC* (i.e. y_{MS}) and a *HMAC* (i.e. x_{MS}), and sends them to the HE together with m in the transaction **T1**. That is,

T1. MS → HE: $IMUI_{MS} \| idSP \| dl \| ts_{MS} \| m \| y_{MS} \| x_{MS}$

where,

$$y_{MS} = h(K_{MS\text{-}SP,\ MS} \|\ m \|\ h(K_{MS\text{-}HE,\ MS}) \| dl),$$

$$x_{MS} = h(IMUI_{MS} \| idSP \| ts_{MS} \| m \|\ K_{MS\text{-}HE,\ MS} \|\ y_{MS})$$

$IMUI_{MS}$ is the MS's identity and *idSP* is the SP's identity, e.g. the URL address of the SP. *dl* defines a time interval with a starting time, t_s, and an ending time, t_e. The reason for introducing *dl* is to prevent the HE from deliberately replaying the signature generation later so as to gain advantage, e.g. the HE may double charge the MS with a replayed signature. The length of *dl*, e.g. $t_e - t_s$, should be reasonably short. ts_{MS} is the MS's time stamp. A fresh time stamp means that the value of the time stamp is between the starting time and the ending time defined in *dl*, i.e. $t_s \le ts_{MS} \le t_e$. y_{MS} is the *HOAC* and includes the secret $K_{MS\text{-}SP,\ MS}$, which is not known by the HE. Thus the HE is unable to forge a valid y_{MS} to the SP. In addition, the HE cannot access to the value of $K_{MS\text{-}SP,\ MS}$ because it is hashed. x_{MS} is the *HMAC* and presents a verifiable message authenticator to the HE for m and y_{MS}. Using the received *HMAC* x_{MS}, the HE can verify the authenticity and integrity of m and y_{MS} received.

Step 2. Upon receipt of **T1**, the HE performs Verification 1:
Verification 1:

1(a) Check that ts_{MS} is fresh, i.e. $t_s \le ts_{MS} \le t_e$.

1(b) Verify $x_{MS} = h(IMUI_{MS,\ HE} \| idSP' \| ts_{MS}' \| m' \|\ K_{MS\text{-}HE,\ HE} \| y_{MS}')$, where $idSP'$, ts_{MS}', m' and y_{MS}' are the items received.

The purpose of Verification 1(a) is for the HE to make sure that **T1** is freshly received. The purpose of Verification 1(b) is to confirm that x_{MS} contains the message m, the secret $K_{MS\text{-}HE,\ MS}$ and the received *HOAC* y_{MS}. As x_{MS} is generated using a keyed hash

Table 1. Definitions of all the items used in the joint-signature scheme

Party	Item	Definition
MS	m	a message originated at the MS for the SP
	$IMUI_{MS}$	the MS's permanent identity stored in the MS
	$K_{MS\text{-}HE,\ MS}$	the session key shared between the MS and the HE, and stored in the MS
	$K_{MS\text{-}SP,\ MS}$	the session key shared between the MS and the SP, and stored in the MS
HE	$IMUI_{MS,\ HE}$ ($= IMUI_{MS}$)	the MS's permanent identity stored in the HE
	$K_{MS\text{-}HE,\ HE}$ ($=K_{MS\text{-}HE,\ MS}$)	the session key shared between the MS and the HE, and stored in the HE
	$pk_{HE} = (e_{HE},\ n_{HE})$	the HE's public RSA key
	$sk_{HE} = (d_{HE},\ n_{HE})$	the HE's private RSA key
	$sign_{HE} = (h(m))^{dHE} \bmod n_{HE}$	the HE's signature over the message m
SP	$K_{MS\text{-}SP,\ SP}$ ($= K_{MS\text{-}SP,\ MS}$)	the session key shared between the MS and the SP, and stored in the SP
	$h(K_{MS\text{-}HE,\ HE})_{SP}$ ($= h(K_{MS\text{-}HE,\ MS})$)	the hash of the session key shared between the MS and the HE, and stored in the SP
	$pk_{HE} = (e_{HE},\ n_{HE})$	the HE's public RSA key

function with the session key $K_{MS\text{-}HE,\ MS}$ (= $K_{MS\text{-}HE,\ HE}$), the verification ensures that the *HMAC* received is authentic and all the items received have not been tampered with. It is worth noting that the HE cannot verify the correctness of the *HOAC* y_{MS} because the HE does not possess the secret $K_{MS\text{-}SP,\ MS}$.

If Verification 1 is positive, the HE produces the joint-signature, denoted as $sign_{HE}$, over the message m, y_{MS} and x_{MS}, and sends it to the SP together with m, x_{MS} and y_{MS} in the transaction **T2**. ts_{HE} is the HE's time stamp.

$$\textbf{T2.}\ \text{HE} \rightarrow \text{SP}: m \parallel x_{MS} \parallel y_{MS} \parallel dl \parallel ts_{HE} \parallel sign_{HE}$$

where

$$sign_{HE} = (h(m \parallel y_{MS} \parallel x_{MS} \parallel dl \parallel ts_{HE}))^{dHE} \bmod n_{HE}$$

If Verification 1 is negative, which means that either the authenticity or integrity of **T1** is not guaranteed, and/or **T1** is not fresh, the HE will refuse to sign the message and terminate the process.

3.3.3 Joint-Signature Verification

Step 3. Upon receipt of **T2**, the SP performs Verifications 2 and 3 defined as follows:

Verification 2:

2(a) Check that ts_{HE} is fresh, i.e. $t_s \leq ts_{HE} \leq t_e$.

2(b) Decrypt the signature with the HE's public key e_{HE}, i.e. $(sign_{HE})^{eHE} = ((h(m \parallel y_{MS} \parallel x_{MS} \parallel dl \parallel ts_{HE}))^{dHE})^{eHE} \bmod n_{HE} = h(m \parallel y_{MS} \parallel x_{MS} \parallel dl \parallel ts_{HE})$, and check if the hash value freshly computed from the received items equals to the decrypted signature value, i.e. $h(m' \parallel y_{MS}' \parallel x_{MS}' \parallel dl' \parallel ts_{HE}') = h(m \parallel y_{MS} \parallel x_{MS} \parallel dl \parallel ts_{HE})$, where m', y_{MS}', x_{MS}', dl' and ts_{HE}' are the items received.

The purpose of Verification 2(a) is for the SP to make sure that **T2** is freshly received, i.e. **T2** is not a replayed message. The purpose of Verification 2(b) is to assure the integrity and non-repudiation of m, y_{MS}, x_{MS}, dl and ts_{HE}. In Verification 3 next, the SP will check the correctness of the *HOAC* y_{MS} signed in $sign_{HE}$.

Verification 3:

Verify $y_{MS} = h(K_{MS\text{-}SP,\ SP} \parallel m' \parallel h(K_{MS\text{-}HE,\ HE})_{SP} \parallel dl')$ where m' and dl' are the items received.

Verification 3 confirms that y_{MS} contains correct $K_{MS\text{-}SP,\ MS}$, m, $h(K_{MS\text{-}HE,\ MS})$ and dl. As $K_{MS\text{-}SP,\ MS}$ (=$K_{MS\text{-}SP,\ SP}$) has been used to generate y_{MS}, y_{MS} is a keyed hash function and thus provides both authenticity and integrity. The inclusion of $K_{MS\text{-}SP,\ MS}$, which is shared between the MS and the SP, assures the SP the origin authentication of the received items, as apart from the SP, only the MS can generate the correct y_{MS} with $K_{MS\text{-}SP,\ MS}$. This, in conjunction with the HE's signature on the items m, y_{MS} and x_{MS}, the SP is protected against repudiation of the invocation of **T1** by the MS. If the result of Verifications 2 or 3 is negative, the joint-signature is invalid.

To summarize, the joint-signature generation is done jointly by the MS generating the *HOAC* (for origin authentication and integrity protection of *m*) and the *HMAC* (for integrity protection of the *HOAC*), and the HE signing the hash of *m*, the *HOAC* and the *HMAC*. The joint-signature verification is done by the SP verifying the signature of the HE and the correctness of the *HOAC*. The *HOAC* value (y_{MS}) generated using a session secret, $K_{MS\text{-}SP,\ MS}$, shared solely between the MS and the SP (the signature verifier), is used to guarantee message origin authentication of *m*. In order to protect the SP against repudiation of message origin by the MS that is also the same secret holder, the *HOAC* is integrity protected using the *HMAC* (x_{MS}) and digitally signed by the HE. Furthermore, in order to protect the SP against false accusation by the MS, or impersonation attacks by the SP or any other entities against the MS, the *HOAC* also imbeds a hashed secret (i.e. $h(K_{MS\text{-}HE,\ MS})$) shared solely between the MS and the HE.

4. ANALYSIS

4.1 Security Analysis

In this subsection, we demonstrate that the joint-signature scheme provides origin authentication, integrity and non-repudiation of origin services.

Proposition 1: The *HMAC* x_{MS} (= $h(IMUI \parallel idSP \parallel ts_{MS} \parallel m \parallel y_{MS} \parallel K_{MS\text{-}HE,\ MS})$) can be generated only by the MS and the HE.

Proof: The *HMAC* x_{MS} is dependent on the knowledge of six data items: *IMUI*, *idSP*, ts_{MS}, message *m*, the session key $K_{MS\text{-}HE,\ MS}$, and the *HOAC* y_{MS}. $K_{MS\text{-}HE,\ MS}$ is only known to the MS and HE [17], so without knowing $K_{MS\text{-}HE,\ MS}$, any third party (e.g. the SP) cannot impersonate the MS to produce the *HMAC* x_{MS} and therefore can not impersonate the MS to take part in a joint-signature generation. Thus only the MS and the HE can generate the correct *HMAC* x_{MS}.

Proposition 2: The *HOAC* y_{MS} (= $h(K_{MS\text{-}SP,\ MS} \parallel m \parallel h(K_{MS\text{-}HE,\ MS}) \parallel dl)$) can be generated only by the MS and the SP.

Proof: The *HOAC* y_{MS} is dependent on the knowledge of message *m*, the session key $K_{MS\text{-}SP,\ MS}$, the hashed session key $h(K_{MS\text{-}HE,\ MS})$ and *dl*. $K_{MS\text{-}SP,\ MS}$ is known only by the MS and the SP [7]. This implies that no other party than the MS and the SP could be able to produce a correct *HOAC* y_{MS} so as to impersonate the MS. Thus only the MS and the SP can generate the correct *HOAC* y_{MS}.

Proposition 3: The joint-signature provides message origin authentication for message *m* generated by the MS.

Proof: Firstly only the MS and the HE are able to produce the correct *HMAC* x_{MS} as discussed in **Proposition 1**. Secondly, only the MS and the SP can produce the correct *HOAC* y_{MS} for the same message *m* as discussed in **Proposition 2**. As a result, the HE and the SP can produce the *HMAC* x_{MS} and the *HOAC* y_{MS}, respectively, but neither of them can produce both of x_{MS} and y_{MS}. In other words, *if and only if* Verifications 1 and 3 are both positive, then it must be the MS who produces both the *HMAC* x_{MS} and the *HOAC* y_{MS} for the message *m*. This guarantees that only the MS could have generated a correctly signed *m*.

Proposition 4: The joint-signature provides integrity and non-repudiation of origin for message *m*.

Proof: The HE's digital signature on the joint-signature provides integrity and non-repudiation of transmission of message *m* from the HE to the SP. In addition, the inclusion of the secret session key $K_{MS\text{-}SP,\ MS}$ in y_{MS} provides origin authentication of the message *m* from the MS to the SP, as discussed in **Proposition 3**.

Therefore the joint-signature can provide integrity and non-repudiation of origin protection for the message *m* to the SP.

With the use of a joint-signature, the MS is able to securely instruct the HE an m-payment related request because the joint-signature ensures that the original generator of message *m* is the MS. In other words, the HE cannot generate a valid joint-signature by itself and therefore can not abuse the trusts given by the MS to send a forged m-payment related request to the SP on the MS's behalf. Therefore the joint-signature scheme can provide a strong accountability service to the asymmetrical m-payment protocols for holding the MS and the HE accountable for their actions/instructions.

4.2 Performance Analysis

In this subsection, we demonstrate that the joint-signature provides an efficient manner for the mobile station to generate a signature.

Computation Load: The joint-signature scheme makes use of the resources in the HE to assist the MS, and its efficiency at the MS is achieved through the use of two hash operations for the generation of the *HMAC* x_{MS} and the *HOAC* y_{MS}. The hash operations can be implemented efficiently in a mobile device.

Communication Load: The joint-signature scheme requires only one transaction sent by the MS to the HE (i.e. **T1**) in order to jointly generate the joint-signature. Any signature scheme, no matter which the MS uses to generate a signature, the MS has to send at least one message to the HE anyway. Thus the joint-signature scheme minimizes the transaction overhead sent by the MS compared with other signature schemes. Message exchanges from the MS in **T1** include seven items: *IMUI*, *idSP*, *dl*, ts_{MS}, *m*, y_{MS} and x_{MS}. The lengths of some of these items are defined by different standards. The length of *IMUI* is 72 bits; the length of *idSP* is 128 bits if it is an IPv6 address; the lengths of y_{MS} and x_{MS} are 160 bits if using SHA-1 [13]. Thus the length of *IMUI*, *idSP*, y_{MS} and x_{MS} is 520 bits. However, the total length of **T1** also depends on the lengths of *dl*, ts_{MS} and *m*, which are session-dependent. Assuming the sending of one SMS message (1120 bits) takes approximately 3 seconds, then sending **T1** will take 3 seconds, if the length of *dl*, ts_{MS} and *m* is equal or smaller than 600 bits. Another way is to use a data call with a 9.6 kbps link, then sending **T1** will take only 0.125 second, if the total length of **T1** is 1120 bits.

One may argue that the computational and communication costs of the joint-signature scheme could be higher if the initialization process is included. We discuss and compare its computational and communication costs with the RSA digital signature method and the proxy/server-aided signature schemes.

• In comparison with a traditional digital signature scheme, the joint-signature scheme requires generating a secret random number, one RSA encryption, and four hash operations online, with the NAETEA protocol for the initialization. All these operations take approximately 0.4ms, but one RSA signature generation takes 10.29ms. (Benchmark refers to [3]. Here we use RSA-1024 public key encryption and RSA-1024 signature generation). Thus the joint-signature scheme is more efficient than a traditional RSA digital signature scheme at the mobile side in terms of computational cost. The communication cost under the joint-signature scheme increases when the NAETEA protocol is included, but this initialization process can be integrated with the entity authentication and key establishment process to reduce the processing load in the joint-signature generation.

• The initialization process is indispensable in the existing proxy/server-aided signature schemes that utilize a third party/proxy to aid signature generation for the original signer. For instance, the undetachable signature scheme [9] requires the MS to generate the undetachable signature function pair, and the server-supported signature scheme, S^3 [1], requires the MS to generate a random secret key, construct the hash chain, and submit the root public key to a Certificate Authority for certification. These schemes as well as the joint-signature scheme require computational and communication costs in the initialization processes. In addition, the initialization process in the joint-signature scheme may be omitted to further reduce the computational and communication overhead if the MS and the SP share a long-term secret (e.g. the MS is a customer who frequently buys tickets from Virgin Trains Online). In this case, the purpose of the initialization process is only for the HE to send a hash value to the SP, and the HE can send the hash value to the SP together with the joint-signature in **T2**. This will require no computational and communication requirement for the MS in the initialization process.

4.3 Comparison with Related Work

The signature schemes [10, 16, 8, 9] make use of a proxy key and are mainly concerned with the delegation of the proxy key(s). Our protocol makes use of the HE's private key and does not require any extra key delegation. Therefore our protocol is more efficient and less complex than these existing protocols.

The protocols [1, 12] also allow a third party to use its private key to sign messages on users' behalf. As the protocol given in [1], called S^3, is more relevant to the topic addressed in this paper, we provide the following detailed comparison between our protocol and the S^3 protocol.

• S^3 protocol requires two signature verifications by using public keys. First, the signature signer needs to verify the certificate on the received secret from the original signer with the public key of the third trusted party. Secondly, the signature verifier needs to verify the S^3 signature with the public key of the signature signer. In our protocol, only the verifier (i.e. the SP) needs to verify the joint-signature with the public key of the signature signer (i.e. the HE), and the signature signer does not need to perform any public key verification. Thus our protocol requires less public key operations than S^3 (see Table 2).

• S^3 protocol requires two additional transactions between the original signer and the signature signer to generate the signature. One transaction is for the original signer to send the message to the signature signer, and another one is for the signature signer to send the signature back to the original signer to produce a NRO (Non-Repudiation of Origin) token [1]. In our protocol, the original signer (i.e. the MS) sends the message to the signature signer (i.e. the HE) with two hash values (i.e. the *HMAC* and the *HOAC*) which are the contents of the joint-signature. Our protocol does not require round-trip transactions between the original signer and the signature signer because the HE is run by the network operator and used to assist the signature generation. The comparison in terms of number of signature verification operation and message sent by the original signer and the signature signer is shown in Table 2. ("No. of Sign. Veri. at sign. Signer" means "Number of signature verification at the signature signer"; "No. of Sign. Veri. at verifier" means "Number of signature verification at the signature verifier"; "No. of msgs sent by original signer" means "Number of messages sent by the original signer"; "No. of msgs sent by sign. signer" means "Number of messages sent by the signature signer".)

Table 2. Comparison of our scheme and S^3 signature scheme

Protocol	Joint-Signature	S^3
No. of Sign. Veri. at sign. signer	0	1
No. of Sign. Veri. at verifier	1	1
No. of msgs sent by original signer	1	2
No. of msgs sent by sign. signer	1	1

In summary, the joint-signature scheme is securer, easier and more efficient to implement.

5. CONCLUSION

In this paper we have presented a novel efficient joint-signature scheme which makes use of a semi-trusted third party. The protocol analysis has confirmed that the protocol can achieve the same security services as those by a traditional digital signature scheme, i.e. origin authentication, integrity and non-repudiation of origin, but offers lighter computational load on one end - mobile users. The performance analysis has shown that the mobile device only need to perform two hash operations, and no public key operations are required. The comparison with related work has demonstrated that our protocol is more efficient. Due to these properties, the joint-signature scheme is well suited to mobile payment applications. For the future work, we intend to formally verify these properties and implement the protocol to demonstrate its efficacy and applicability.

6. REFERENCES

[1] Asokan, N., Tsudik, G. and Waidner, M., Server-supported signatures, Journal of Computer Security 5, 1, 91-108, 1997.

[2] Chen, T.-S., Huang , G.-S., Liu, T.-P., and Chung, Y.-F., Digital signature scheme resulted from identification protocol by elliptic curve cryptosystem, In Proceedings of the 2002 IEEE TENCON'02, Vol. 1, pp. 192-195, 2002.

[3] Dai, W., Crypto++ 4.0 Benchmarks, 2000, http://www.eskimo.com/~weidai/benchmarks.html.

[4] Diffie, W. and Hellman, M. E., New directions in cryptography, IEEE Transactions on Information Theory, IT-22(6): 644-654, 1976.

[5] Ding, X., Mazzocchi, D. and Tsudik, G., Experimenting with server-aided signatures, In Proceedings of the Network and Distributed Systems Security Symposium, 2002.

[6] ElGamal, T., A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory, Vol. IT-31, pp. 469-472, July, 1985.

[7] He, L. and Zhang, N., An asymmetric authentication protocol for M-Commerce applications, In Proceedings of the eighth IEEE ISCC'2003, Vol. 1, pp. 244-250, 2003.

[8] Kim, S., Park, S. and Won, D., Proxy signatures, revisited, In Proceedings of ICICS97, LNCS 1334, Springer Verlag, pp. 223-232, 1997.

[9] Kotzanikolaou, P., Burmester, M. and Chrissikopoulos, V., Secure transactions with mobile agents in hostile environments, LNCS 1841, Springer Verlag, pp. 289-297, 2000.

[10] Mambo, M., Usuda, K. and Okamoto, E., Proxy signatures: delegation of the power to sign messages, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E79-A, No. 9, pp. 1338-1354, 1996.

[11] Matsumoto, T., Imai, H., Laih, C. S. and Yen, S. M., On verifiable implicit asking protocols for RSA computation, In Proceedings of Auscrypt'92, LNCS 718, Springer Verlag, pp. 296-307, 1993.

[12] Matsumoto, T., Kato, K. and Imai, H., Speeding up computation with insecrue auxiliary devices, In Proceedings of Crypto'88, LNCS 403, Springer Verlag, pp. 497-506, 1989.

[13] National Institute of Standards and Technology, Secure hash standard, Federal Information Processing Standards Publication 180-1, 1995.

[14] Romao, A. and Silva, M. M., Proxy certificates: a mechanism for delegating digital signature power to mobile agents, In Proceedings of the Workshop on Agents in Electronic Commerce, pp. 131-140, 1999.

[15] Schneier, B., Applied cryptography: protocols, algorithms, and source code in C, John Wiley & Sons, Inc, 1996.

[16] Sun, H.-M., Lee, N.-Y. and Hwang, T., Threshold proxy signatures, In IEE Proceedings -Computers and Digital Techniques, Vol. 146, No. 5, pp. 259-263, 1999.

[17] Technical Specification 3GPP, 3G security; security architecture, http//www.3gpp.org, 2000.