Generating Digital Signatures on Mobile Devices

Yu Lei
College of Computer Science
Zhejiang University
Hangzhou, 310027, P.R.China
leiyu@zju.edu.cn

Deren Chen
College of Computer Science
Zhejiang University
Hangzhou, 310027, P.R.China
drchen@cs.zj.edu.cn

Zhongding Jiang
State Key Lab. of CAD&CG
Zhejiang University
Hangzhou, 310027, P.R.China
zdjiang@cad.zju.edu.cn

Abstract

With the explosion of the mobile communication market, more and more handheld devices act as clients in the Internet. People use these devices to purchase books, play games, receive emails, etc. For protecting privacies, such applications should integrate digital signature schemes. Since handheld devices have poor computational capabilities and limited battery life, traditional computation intensive digital signature protocols that are based on asymmetric cryptographic algorithms are not suitable for mobile devices. In this paper, we propose a Server Based Signature (SBS) scheme for mobile devices. Besides achieving the same security level of the traditional digital signature protocols, the SBS scheme also: 1) reduces the computation complexity on the mobile devices; 2) reduces the communication consumption between signer and verifier. Application results show that our scheme is very useful for mobile communication systems.

1. Introduction

With the rapid explosion of the Internet, people's daily activities heavily rely on it. Huge of commercial transactions are conducted in the Internet. Thanks to the rapid development of mobile communication technologies, people can now use cell-phones, palmtops and PDAs to access the Internet anywhere and anytime. However, many mobile communication applications are faced with some common problems: privacy and security. Such applications include:

- mobile payment system
- remote walk-through system
- electronic wallet
- e-ticket system
- image authenticating and exchanging

Digital signature can be represented as a secure base in such applications because it provides authentication, data integrity and non-reputation cryptographic services. Traditional digital signature schemes were based on asymmetric cryptographic techniques which made the signature computation very expensive. In spite of handheld devices can come in many shapes and be used for different purposes, they have the common limitations: 1) limited computational capability, 2) short battery life. If the traditional asymmetric cryptographic computations are executed on mobile devices, those devices would be blocked for a period of time, and drain batteries quickly.

There are many proposed digital signature schemes in literature. According to their bases, these schemes can be classified into two general categories: message digest based schemes and message recovery based schemes. In message digest based digital signature scheme, the original message is first mapped to a checksum by a one-way function. Then this checksum (message digest) is used to generate a digital signature. The checksum used here is to provide data integrity. In message recovery based scheme, the receiver can recover the original message from the received signature. These are done by message redundancy scheme.

There are also some work on digital signatures for mobile devices. Asokan *et al.* [1] proposed Server-Supported Signature scheme for mobile communication. Their work employed a one-way function and traditional digital signature scheme. Signature servers were responsible for generating signature tokens and certification authorities to verify these tokens. Therefore the scheme's robustness depends on the reliability of those servers. Based on the work of Asokan *et al.* [1], Ding *et al.* [2] presented a modified digital signature scheme, called Server Aided Signature. In this scheme, users are involved in the generation of the signature token, giving the on-line feature and defending the DOS attack. Unlike the traditional digital signatures which often use the pair public/secret keys to generate non-reputation signature token, [1] and [2] use one way hash function to generate sender's secret key and use this key to produce non-reputation signature token. This is due to the fact that generation of robust pair public/secret keys is computation intensive for handheld devices. But in their schemes, asymmetric cryptographic computation is still expensive in clients because during the period of verification of digital signature, clients should decrypt the encrypted signature token to verify the associated content added by signature servers.

In this paper we present a Server Based Signature (SBS) digital signature scheme for mobile devices, Our scheme aims to:

- help mobile devices to generate digital signature efficiently.
- reduce communication consumption between partners during signature process.
- provide robust security as the traditional digital signature schemes have.

Using an adopted signature way in [5], we avoid the expensive computation when user verifying signature generated from server without losing any security strength. Thus, our scheme is more efficient than [1] and [2].

The rest of this paper is organized as follows. Section 2 describes the details of our Server Based Signature (SBS) scheme. In Section 3, we give the security and efficiency analysis of our scheme. In section 4, we show one application scenario. Finally we draw the conclusions.

2. Server Based Signature

There are various classes of non-reputation services. We have special interests in two classes: Non-Repudiation of Sender (NRS) and Non-Repudiation of Receiver (NRR). NRS guarantees that the sender of a message cannot later deny having sent that message. NRR guarantees that the receiver of a message cannot deny having received that message.

In subsection 2.1, we present NRS signature as the base of SBS. Section 2.2 gives the whole SBS system with NRS and NRR. The SBS working process is shown in Figure 1 , excluding the generation of tokens.

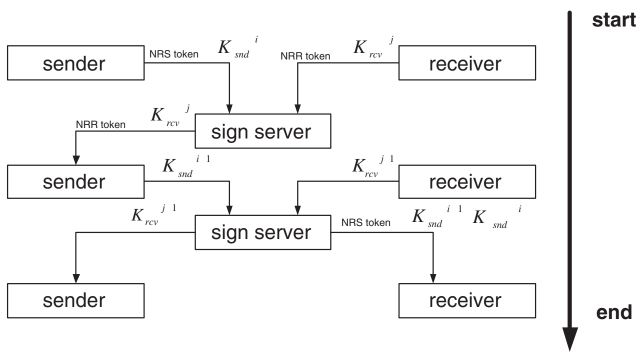

Figure 1. SBS workflow

2.1. SBS's NRS Signature Protocol

All participants agree on a one-way collision-resistant hash function, examples SHA1 or MD5 [3]. Furthermore, sender and receiver should take a *personal* hash function pair $h_{snd}()$ and $h_{rcv}()$. We call these hash functions as keyed hash functions. This can easily be done by inserting sender's or receiver's identify as an argument of these hash functions. $h^i_{snd}()$ and $h^i_{rcv}()$ denote the keyed hash functions linked to the identity with the index i (i times hash operation).

Each user generates a secret key K_u (u represents the user). Using K_u as original input, user can construct one hash chain K_u^0, K_u^1, …, K_u^n, as $K_u^0 = K_u, K_u^i = h_u^i(K_u) = h_u(K_u^{i-1})$. $PK_u = K_u^n$ is considered as the user's public key. Assuming the sender wants to send message x to the receiver. With notations defined above, we illustrate the digital signature process as following.

Step 1: The sender should first generate its public key PK_{snd}, then selects a sign server (S). Finally, the sender submits the user's identify, maximum number of signatures n, public root key PK_{snd} and S to a Certification Authorities (CA) to achieve one certificate. The form of the certificate is as following:

$$Cert_{snd} = SK_{CA}(Sender, n, PK_{snd}, S)$$

SK_{CA} represents the CA's secret key. After the certification is finished, the result can be made public via some directory services, such as LDAP [6] .

Step 2: The sign server (S) generates two large primes p and q and let $m = p \times q$, $\phi(m) = (p-1) \times (q-1)$. A number d is also selected as $e \times d = 1 \bmod \phi(m)$, where $e = 3$. S keeps (d, p, q) as its secret key and publishes(e, m).

Step 3: If the sender wants to generate a digital signature for content x, it first randomly chooses two number r and v, r , $v \in Z_n^*$. Then it computes $\delta = r^e h(x)(v^2 + 1) \bmod m$ and sends (δ, K^i_{snd}, i) to S. Then S randomly chooses a positive number $z, z < m$ and sends it back to the sender. After receiving number z, the sender randomly selects another integer r' and computes $b, b = r \times r'$. Finally, the sender gets η, where $\eta = b^e \times (v - z)$ and sends it back to S.

Step 4: S computes $\gamma = \eta^{-1} \bmod m$ and $t = h(Cert_{snd})^d(\delta(z^2 + 1)\eta^{-2})^{2d} \bmod m$, then sends (γ, t) to the sender. Upon receiving (γ, t), the sender computes $c = (vz + 1) \times \gamma \times b^e = (vz + 1)(v - z)^{-1} \bmod n$, and $s = t \times r^2 \times {r'}^4 \bmod m$. The triple$(Cert_{snd}, c, s)$ is the signature of message x. Because $s^e \equiv h(Cert_{snd})h(x)^2(c^2 + 1)^2 \bmod m$ [5], so everyone can verify the signature $(Cert_{snd}, c, s)$ of message x efficiently. After verifying the signature, the sender sends $(Cert_{snd}, c, s)$ and K^{i-1}_{snd} back to the sign server.

S verifies the signature and $K_{snd}^{i} = h(K_{snd}^{i-1})$. If all results are correct, S sends message x with its signature to the receiver. The $(Cert_{snd}, c, s)$, K_{snd}^{i-1} and K_{snd}^{i} consist the NRS token.

Step 1 to 4 show how SBS's NRS subsystem works. The detailed non-reputation and security discussions will be discussed in section 3.

2.2. SBS's NRS and NRR Signature Protocol

We have solved the NRS problem in previous subsection. This subsection will solve the Non-Repudiation of Receipt (NRR) problem. Based on SBS's NRS scheme, it is easy to extend it for solving the NRR problem. As those processes described from step 1 to step 4, the receiver also generates a signature $(Cert_{rcv}, c', s')$. We denote $(Cert_{rcv}, c', s')$ as a NRR token. During step 4, before the sender sends K_{snd}^{i-1} to the sign server, the receiver should send this NRR token with its public key K_{rcv}^{j} to the sender. Then the sender sends its K_{snd}^{i-1} to the sign server after verifying the NRR token. After the sign server received both K_{snd}^{i-1} and K_{rcv}^{j-1}, it sends the sender's digital signature $((Cert_{snd}, c, s), i, K_{snd}^{i}, K_{snd}^{i-1})$ to the receiver and K_{rcv}^{j-1} to the sender. Thus the signature process of SBS is finished.

2.3. Repudiation Analysis

In case sender claimed it didn't send the message signatured by an NRS token, receiver could provide the token, message, certificate and the digital signature to an arbiter. The arbiter first verifies the certificate $(Cert_{snd})$ which was certified by the CA with the sender's identify, public key, sign server's identify, etc. Then it checks the signature $(Cert_{snd}, c, s)$ using the formula $s^e \equiv h(Cert_{snd})h(x)^2(c^2+1)^2 \bmod m$. Finally, the arbiter should determine whether the sender's public key $PK_u = K_u^n$ can be derived from key K_u^{i-1}, K_u^i with the public one-way collision-resistant hash function. If all the results are correct, the arbiter can believe that the sender have sent the message to the receiver. With these successful checking results, if the sender still wanted to claim that it didn't send the message, it should provide the evidences show that:

1. CA cheated: In our scheme, every sender should first register its public key associated with a sign server. If sender wants to claim CA is cheated, it should give a certificate with a different public key. Thus when this certificate is shown, the CA must give the answer why this happened. CA can prove its innocent by showing people the register contract (the process of certification).
2. sign server cheated: To prove the sign server is cheated, the sender should give another signature $(Cert_{snd}, c, s)$ and K_{snd}^{i-1} which can sign message x. We all know, if sender didn't provide K_{snd}^{i-1}, it is computational infeasible to derive K_{snd}^{i-1} from K_{snd}^{i} because of the one-way speciality of the hash function $h()$. The signature $(Cert_{snd}, c, s)$ is generated from message x with SHA and RSA [4], so making a fake signature is computational infeasible.

As shown in Figure 1, the sign server must be trustworthy to avoid NRR and NRS problems. Like well-established asymmetric techniques, we only use verifiable servers (whose cheating can be proven by an arbiter). The analysis of NRR is the same as NRS. Relying on the collision-resistant one-way speciality of the hash function, we can easily prevent the repudiations in both sender and receiver.

3. Security and Efficiency Analysis

3.1. Security

The server based signature (SBS) scheme can meet the same security level as the traditional digital signature schemes. If the attacker can break our signature scheme, he can also hack the SHA and RSA cryptographic algorithms. In most cases, the attacks to the digital signature are equivalent to forging a signature or making a fake signature.

Forging signature is one common attacking approach. In this case, the attacker will try to forge an SBS's signature $(Cert_u, c, s)$. The attacker may have valid signature or not and tries to derive some forged signatures. In our scheme, all of these attacks will be failed. Firstly, we assume the attacker has the valid signature. To pass the signature verification $s^e \equiv h(Cert_u)h(x)^2(c^2+1)^2 \bmod m$ successfully, the attacker should get an s', where $s' \equiv h(Cert_u^d)h(x)^{2d}(c^2+1)^{2d} \bmod m$, given values $h(Cert_u)$, $h(x)$ and c. However, it is computational infeasible to deduce d from e and n without the large prime number p and q according to RSA[4] algorithm. Due to the same reason, the attacker also can't deduce c, where $c^2 \equiv (s^e \times h(Cert_(u))^{-1} \times h(x)^{-2})^{\frac{1}{2}} - 1 \bmod n$. Thus, the attacker cannot forge a signature even he has a valid signature. On the other hand, if the attacker didn't have a valid signature $(Cert_u, c, s)$, it is more difficult to deduce such signatures. Thus forging an SBS signature is as hard as traditional digital signature schemes, such as DSS [7].

To make a fake signature, the attacker should try to find one K_u^{i-1}, where $h_u(K_u^{i-1}) = K_u^i$, K_u^i is published by the user. However, finding such K_u^{i-1} means a successful attack on the one-way characteristics of the hash function $h_u()$, example as SHA. As we know, such attack is computational infeasible.

From the above security analysis, we can claim that the underlying theories of our SBS scheme were as sound as those of traditional digital signature schemes.

3.2. Efficiency

The whole cost of our SBS scheme can be divided into two parts:

1. Computation cost. In our scheme, the computation consists of:
   (a) sign server computation: The server should first generate a pair of RSA keys, then publish them through some secure directory services. Finally, it uses the asymmetric cryptograph algorithm to produce the digital signature on the given message x.
   (b) user computation: User should first compute its hash-chain values by some special input pre-images. And in step 4 of our scheme, user must verify the signature generated by the sign server.
2. Communication cost. To sign a message, the user relies on the sign server to execute most computation of generating digital signature. So it needs the communication channels between the user and the sign server during signaturing process.

Comparing to the traditional digital signature scheme [7], the overhead of network communication, user's public root key generation and verification of signature token are extra cost. However, our scheme is more suitable for weak power handheld devices to produce secure digital signatures. In our scheme, the users (handheld devices) only need to compute hash-chain values and verify the signatures. The hash-chain computation is as same as the Server Supported Signature [1]. The signature verification operation of our scheme is more efficient than that in [1].

For decrypting a km bits message with a k bit key, RSA is m times complex than that of computing a k bits modular exponential operations. With a modulus n, the computation for a modular exponential operation is taken as $0.3246|n|$[1] modular multiplications [8]. As we have shown, our algorithm of generating the digital signature only need several modular multiplications (nearly 20), but the scheme in [1] need m times modular exponential operations. With a normal key length 1024, it is clearly that Server Based Signature consumes less power than the Server Supported Signature does.

4. Application

Our SBS scheme can be used in many systems, such as mobile code verification, mobile payment, remote education and image authenticating, etc. we have integrated it into one digital artwork authenticating and exchanging system. The block diagram of our system is not shown here due to page limit.

Alice (author) published her digital artwork in the artwork web server. To protect the copyrights of these digital products, she may use some copyright protecting techniques, such as watermark. Bob (consumer) first browses the products published in the web server. If he is interested in some products and wants to purchase them, he should first pay for the products through the payment server. After Bob has done these above processes, Alice checks the payment, then signs the digital artwork, and sends them to Bob through the sign server.

5. Conclusion

In this paper we propose one digital signature scheme for mobile devices, including palmtops, cell phones and PDAs. Due to the poor computational power and limited battery life cycle, these devices are not suitable for running traditional asymmetric cryptographic algorithms. Our Server Based Signature (SBS) scheme consumes less computational resource on the client sides (mobile devices) and is more practical than traditional schemes. The security analysis of our scheme also proves that it can achieve the same security level as the traditional ones. Our SBS scheme can be used in many systems, such as mobile code verification, mobile payment, remote education and image authenticating and so on.

1 $|n|$ mean the bit length of n

References

[1] N. Asokan, G. Tsudik, M. Waidner. Server-supported signatures. *Journal of Computer Security*, Volume 5, Issue 1, pages 91–108, January 1997.

[2] Xuhua Ding, Daniele Mazzocchi, Gene Tsudik. Experimenting with Server-Aided Signatures. In *Proceedings of Network and Distributed System Security Symposium (NDSS'2002)*, San Diego, 2002.

[3] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, Inc. 1996.

[4] Public Key Cryptography Standards(PKCS), No.1, *RSA Encryption standard*, http://www.rsasecurity.com/rsalabs/pkcs

[5] Hung-Yu Chien, Jinn-Ke Jan, Yuh-Min Tseng. RSA-based partially blind signature with low computation. In *Proceedings of International Conference on Parallel and Distributed Systems (ICPADS'2001)*, 2001.

[6] S. Boeyen, T. Hows, P. Richard. Internet x.509 public key infrastructure operational protocols-LDAPv2. *RFC 2559*, 1999.

[7] National Institute for Standards and Technology. Digital Signature Standard (DSS). *Technical Report 169*, August 30, 1991.

[8] V. Dimitrov, T. Cooklev. Two Algorithms for Modular Exponentiation Using Nonstandard Arithmetic. *IEICE Trans. Fundamentals*, Volume E78-A, pages 82–87, 1995.