Nonrepudiable Proxy Signature Schemes

Kan Zhang

Cambridge University Computer Laboratory
Pembroke Street, Cambridge CB2 3QG, UK
Email: kz200@cl.cam.ac.uk

Abstract. Delegation of rights is a common practice in the real world. Proxy signature schemes have been invented to delegate signing capability efficiently and transparently. Existing proxy signature schemes don't support nonrepudiation. Nonrepudiation means signature signer, both the original and proxy signers, cannot falsely deny later that he generated a signature. In practice, it is important and, sometimes, necessary to have the capability to decide who is the actual signer of a proxy signature for internal auditing purpose or when there is abuse of signing capability. In this paper, we show how to add nonrepudiation to existing proxy signature schemes. Nonrepudiation is achieved through a partially blind signature key generation protocol adapted from a fully blind signature scheme. The new nonrepudiable proxy signature scheme fits in the same general framework as existing ones. Therefore, the desirable properties of existing schemes can also be said about our new scheme. In addition, our new scheme has some other advantages, such as proxy signature key generation and updating using insecure channels. Our approach can also be applied to other ElGamal-like proxy signature schemes.

1 Introduction

As the Internet is used more and more for business, adequate security mechanisms in the electronic world are needed to replace established practice in the paper-based world. Digital signature scheme is one of them. Digital signatures have been invented as a counterpart of handwritten signatures. A digital signature provides the proof of authenticity of document content and its originator as handwritten signature does. However, while basic digital signature schemes are able to provide most of the functionalities of personal signature, they are less than ideal for institutional purpose.

For institutions, seals are often used for signing on behalf of institutions. An institutional seal represents the institution, not the person who has authority to use the seal. A major difference between signing by hand and by seal is that seal is transferable. This transferability allows efficient and flexible delegation of signing capability, and therefore reduces costs. For example, when a manager of a company goes on holiday, he may delegate his company seal to his deputy to sign on behalf of the company. The requirement here is transparent delegation, which

means the customers of a company are not affected. The signatures presented to them are the same as before and they can verify the signatures using the same process. It will be economically infeasible for a big company to notify all its customers each time there is a personnel change in the company.

Unfortunately, basic digital signature schemes, like signing by hand, are not transferable. Digital signature schemes rely on a secret signature key which only the certified person knows. If this secret key is delegated to another person, it can no longer be identified with that person and, hence, the underlying assumption of digital signature schemes is broken.

What we need is a way to delegate signing capability without revealing the original signer's secret signature key to the proxy signer such that signature recipients can verify proxy signatures signed by a proxy signer using only the original signer's public key with minimum changes to the verification process.

A hardware-based solution is to put the company signature key in a temper-proof hardware, e.g. a smartcard, and delegate it to a proxy signer. The underlying assumption is that the hardware is temper-proof, which is not necessarily true [1]. Another drawback is that you cannot tell the differences between signatures signed by a proxy and those signed by the original signer since they are all signed by the same smartcard (same key). Therefore, when there is internal dispute, it is difficult to resolve.

In practice, it is important, and sometimes necessary, to have the ability to know who is the actual signer of a proxy signature for internal auditing purpose or when there is abuse of signing capability. We call this property *Nonrepudiation*. Nonrepudiation means signature signers, both the original and proxy signers, cannot falsely deny later that he generated a signature.

In this paper, we are interested in software-based solutions without assuming temper-proofness of hardware. Existing software-based solutions do not support nonrepudiation among original and proxy signers. We show how to achieve this property. Using our nonrepudiable proxy signature key generation protocols given below, the original signer cannot derive proxy signer's signature key and vice versa. Therefore, signature signers can be uniquely identified. In the meantime, signatures computed using a proxy signature key can be verified using the original signer's public key. Hence, the verifier can be convinced that the signatures are from an authorised party of the original signer. In addition, our new scheme has some other advantages, such as proxy signature key generation and updating using insecure channels. Our approach can also be applied to other ElGamal-like proxy signature schemes.

The paper is organized as follows. In Section 2, we discuss various software-based approaches to delegation of signing capability. In Section 3, a brief description of recent work is given and problems identified. Solutions are given in Section 4. We discuss our approach in Section 5. Properties and generalisations of our scheme are given in Section 6. Finally, we draw our conclusions in Section 7. In the Appendix, two signature schemes related to our work are briefly described.

2 Delegation of signing capability

Delegation of rights has been studied for various applications before, e.g. proxy-based user authentication [20, 14]. Since digital signature schemes ensure both message authentication and identity of its creator, they are often used as a method to certify intended delegations [20, 14]. There are other special digital signature schemes for delegating verification capability, e.g. undeniable signatures [8, 5], designated confirmer signatures [6, 18].

In this paper, we are interested in delegating signing capability. While delegation of signing capability as a tool may have broad applications in various areas, we confine ourselves to the problem of how to delegate signing capability to a proxy signer. In [13], delegations of signing rights are classified into three types, i.e., *full delegation, delegation by warrant* and *partial delegation.*

In full delegation, a proxy signer is given the same secret key x that the original signer has, so that the proxy signer can create the same signature as the original signer does. As shown above, in this approach the signature key cannot be identified with the person it was originally certified to. In many situations, the original signer has to update his signing key afterwards, which is not cost-effective for business applications. Moreover, since signatures created by proxy signers are indistinguishable from those created by the original signer, there is no deterrent to possible abusing.

Delegation by warrant is implemented by using a warrant, which certifies that the proxy is trusted to sign on behalf of the original signer under certain conditions. Delegation by warrant is performed by consecutive execution of signing of the public key signature scheme, and this type of delegation has appeared in the literature, e.g. [20, 14]. There are two types of signature schemes in this approach. In the first type, a warrant is composed of a message part and an original signer's signature for a public key of the proxy. In the second type, a warrant is composed of a message part and an original signer's signature for a newly generated public key. These two types correspond to two classes of proxies in [14], called a delegate proxy and a bearer proxy, respectively. Delegation by warrant can be implemented using ordinary signature schemes without modification. However, we need to do two signature verifications to verify one proxy signature, and the verification process is changed from one-step to two-step.

In partial delegation, a proxy signature key created from the original signer's signature key is given to a proxy signer. Signatures created using the proxy signature key can be verified using a modified verification equation in such a way that the verifier can be convinced that the signatures are from an authorised party of the original signer. The verifier needs to verify one signature only and only the public key of the original signer is required. Signatures created by proxy signers are distinguishable from those by the original signer. Furthermore, signatures signed by the original signer can be treated as a special case of proxy signatures and the same verification process can be used for both original and proxy signatures. Therefore, compared with delegation by warrant, partial delegation is more efficient and transparent.

In this paper, we are interested in methods for partial delegation of signing capability.

3 Proxy Signature Scheme by Mambo, Usuda and Okamoto

Recently, Mambo, Usuda and Okamoto proposed a proxy signature scheme based on discrete logarithm for partial delegation of signing capability [13]. The proposed scheme, hereafter referred to as MUO scheme, has many desirable properties. In this section, we give a brief description of the MUO scheme. Interested readers are referred to [13] for details.

The system parameters consist of a prime p, a prime factor q of $p-1$, and an element $g \in Z_p^*$ of order q. The original signer's private key is a random element $x \in Z_q$, and the corresponding public key is $y = g^x \pmod{p}$. MUO scheme uses the following protocol.

- Step 1. (Proxy key generation) An original signer selects $k \in Z_q$ at random and computes $r = g^k \bmod p$. After that, he calculates proxy signature key $s = x + kr \bmod q$.
- Step 2. (Proxy key delivery) The original signer gives (s, r) to a proxy signer in a secure way.
- Step 3. (Proxy key verification) The proxy signer checks that

$$g^s = yr^r \pmod{p} \tag{1}$$

  If (s, r) passes this congruence, she accepts it as a valid proxy signature key. Otherwise, she rejects it and requests another one, or she can simply stops the protocol.
- Step 4. (Signing by the proxy signer) When the proxy signer signs a message m on behalf of the original signer, she computes a signature S_p using the original signature scheme and s as the secret signature key. The proxy signature is (S_p, r).
- Step 5. (Verification of proxy signature) The verification of proxy signatures is carried out using the same verification equation as in the original signature scheme except for replacing y with $y' = yr^r \pmod{p}$.

The following equation can be used as an alternative to equation (1).

$$g^s = y^r r \pmod{p} \tag{2}$$

Equation (1) and (2) correspond to a special case (message m equals 1) of Yen-Laih [21] and Nyberg-Rueppel [17] signature schemes, respectively. The authors of MUO scheme stated the following properties.

- *Unforgeability* Only the original signer and the designated proxy signer can create a valid proxy signature.

- *Proxy signer's deviation* A proxy signer cannot create a valid proxy signature not detected as a proxy signature.
- *Verifiability* From proxy signatures, a verifier can be convinced of the original signer's agreement on the signed message.
- *Distinguishability* Valid proxy signatures are distinguishable from normal self-signing signatures.
- *Identifiability* An original signer can determine the proxy signer's identity from a proxy signature.
- *Undeniability* Once a proxy signer creates a valid proxy signature for an original signer, it is not disavowed even by the proxy signer.

In the MUO scheme, while a valid proxy signature cannot be disavowed by the original or the proxy signer, it is impossible to decide who is the actual signer given a proxy signature since both the original signer and the proxy signer know the proxy signature key. Therefore, there is no nonrepudiation among original and proxy signers. In this paper, we show how to achieve this property. Another drawback of MUO scheme is that it requires a secure channel to pass the proxy signature key from the original signer to the proxy signer. Anyone who can intercept this proxy signature key can impersonate the proxy signer. In the next section, we propose a nonrepudiable proxy signature scheme which overcomes these shortcomings.

4 Nonrepudiable proxy signature scheme

To achieve nonrepudiation, we use blind signature schemes to generate proxy signature keys. However, direct applying existing blind signature schemes [7, 12] results in fully blind signatures such that given a proxy signature the original signer has no idea about the identity of its proxy signer. Therefore, we have to adapt fully blind signature schemes to partially blind signature schemes in which only the proxy signer knows the secret proxy signature key. In the meantime, the original signer or a third-party observer knows the corresponding public key. In this way, we are able to achieve nonrepudiation while retaining identifiability. We modify the blind Nyberg-Rueppel scheme [7], corresponding to Eq (2) in the MUO scheme, as an example to show the general approach. The protocol goes as follows.

1. Original signer selects $\tilde{k} \in Z_q$, computes $\tilde{r} = g^{\tilde{k}} (\mathrm{mod}\, p)$, and sends $\tilde{r}$ to Proxy signer.
2. (a) Proxy signer randomly selects $\alpha \in Z_q$, computes
$$r = g^{\alpha} \tilde{r} (\mathrm{mod}\, p) \tag{3}$$
   (b) Proxy signer checks whether $r \in Z_q^*$. If this is not the case, he goes back to step (a). Otherwise, he sends r to Original signer.
3. Original signer computes $\tilde{s} = rx + \tilde{k} (\mathrm{mod}\, q)$ and forwards $\tilde{s}$ to Proxy signer.

4. Proxy signer computes $s = \tilde{s} + \alpha \pmod q$, and check if the following equality holds:
$$g^s = y^r r \tag{4}$$
If it holds, Proxy signer accepts s as a valid proxy signature key from Original signer.
5. Proxy signature generation and verification are the same as in MUO scheme.

5 Discussion

It is obvious that our proxy signature key generation process is an adapted version of the blind Nyberg-Rueppel scheme proposed by Camenisch et al. [7]. In their scheme [7], the generated blind signature pair (r, s) (see Appendix) is statistically independent from the original signer's view of an execution of the protocol, which means the original signer cannot get any information about the signature pair he blindly signed. This is the blindness property in Chaum's original sense [3, 4], which we refer to as *full blindness* in this paper. While fully blind signature schemes may find useful applications in digital cash, they are not ideal for generating proxy signature keys. In proxy signature schemes, we need a kind of partially blind signature scheme which prevents the signer from knowing part of the blindly signed signature pair. This secret part can be used as the proxy signature key. Yet the original signer should know enough information about the blindly signed signature pair to prevent the proxy signer from generating other valid signature pairs and, therefore, the original signer can identify the proxy signer from the proxy signatures he generated.

In [7] (see Appendix), two blinding factors, $\alpha \in Z_q$ and $\beta \in Z_q^*$, are used. It is proved that given a valid signature pair (r, s) and any view V representing the original signer's complete view of an execution of the protocol, i.e. his random coin tosses and all exchanged values, there exists a unique pair of blinding factors $\alpha \in Z_q$ and $\beta \in Z_q^*$. Since the proxy signer chooses the blinding factors α and β randomly, the full blindness of the signature scheme follows.

To change the fully blind signature scheme into a partial one, we can use a fixed public known value for β. For simplicity, we choose $\beta = 1 (\mathrm{mod}\, q)$. In this case, r, and, therefore mg^α are known to the original signer. To further make α and, therefore, s unique, we need to let message m publicly known. For simplicity, we choose $m = 1$. This reduces the underlying Nyberg-Rueppel scheme (see Appendix) to the one used in MUO scheme (Eq 2). And the original blind Nyberg-Rueppel scheme becomes the one we use in our proxy signature key generation.

In [10], Franklin and Yung identified three forms of blinding:

- **signature with blind verification** prevents the signer from later recognising the signature, without necessarily hiding the message from him.
- **signature with blind message** prevents the signer from later recognising the message being signed, without necessarily hiding the signature from him.

– **fully blind signature** combines blind verification and blind message in a single signature scheme.

Our partially blind signature scheme derived from [7] doesn't belong to any of these categories. In our scheme, message $m = 1$ and part of the signature, i.e. r, are public. The original signer can uniquely recognise a proxy signer from r. What is hiding from the original signer is the s corresponding to an identifiable r.

6 Properties and Generalisations

The difficulty of finding the corresponding proxy signature key s by the original signer knowing r is equivalent to solving the discrete logarithm problem (Eq (3) and Eq (4)). Hence, it is computationally infeasible for the original signer to produce a proxy signature (s', r'), where $r' = r$. On the other hand, the impossibility of computing x, $\tilde{k}$ or forging another valid signature that satisfies Eq (4) by the proxy signer is guaranteed by the underlying signature scheme [17]. Note that message substitution attack is not feasible in our scheme since we require a constant message $m = 1$. For more detailed discussion on the security of ordinary ElGamal-like signature schemes, please see [9, 21, 17, 2, 16].

It is apparent that our scheme and the MUO scheme belong to the same general framework. The properties that the MUO scheme has, for example, the properties listed in Section 3, are also true with our scheme. The efficiency gained by performing just one verification compared with two in the delegation by warrant can also be said about our scheme. Signatures signed using the original signature key can also be verified using the same Eq (4) by letting $r = 1$, which means the same verification process can be used. In addition, the proxy revocation mechanism proposed in [13] can also be applied to our scheme.

The difference between our scheme and MUO scheme is that we use a partially blind signature protocol to generate the proxy signature key. This allows us to achieve nonrepudiable proxy signatures, i.e. only the proxy signer is able to compute a proxy signature verifiable using his proxy signature key.

Note that although the original signer can go through the protocol himself and compute an arbitrary proxy signature key, he is unable to impersonate any proxy signers since α is chosen by proxy signers secretly and independently.

Another merit of our protocol is that all communications between an original signer and a proxy signer need not be kept secret. An attacker knowing all the messages passed between an original signer and a proxy signer is at no advantage over either the original signer or the proxy signer. It doesn't help even the attacker may intercept many proxy signature key generation sessions.

Whereas in the MUO scheme, the proxy signature key has to be sent to the proxy in a secure way. Anyone who manages to get the key can impersonate the proxy signer. This could be a weak point. In an attempt to overcome this

drawback, the authors provide a proxy key updating protocol using insecure channels [13].

A basic requirement of key updating protocols is to provide forward and backward security, which means knowing a current session key will not lead to the compromise of past and future session keys. Unfortunately, using their key updating protocol, all the proxy signature keys are linked. Once an attacker gets any of the used proxy signature keys, he can compute all the past and future proxy signature keys using the information he collects from the insecure channel.

Our scheme doesn't suffer from this drawback. When updating proxy signature keys, the original signer and the proxy signer can simply go through the key generation protocol again using the insecure channel. If they follow the protocol specification, the generated proxy signature keys should be independent from each other.

We have shown our approach through an adapted blind Nyberg-Rueppel scheme. Since all the ElGamal-like signature and blind signature schemes share the same general construction [11, 12], we are able to apply our approach to other ElGamal-like proxy signature schemes. Details will be given in the full paper.

7 Conclusion

We have shown how to add nonrepudiation to existing proxy signature schemes. Nonrepudiation means the signature signer, both the original and proxy signers, cannot falsely deny later that he generated a signature. This property is achieved through a partially blind signature key generation protocol adapted from a fully blind signature scheme. The new nonrepudiable proxy signature scheme fits in the same general framework as existing ones. Therefore, the desirable properties of existing schemes can also be said about our new scheme. In addition, our new scheme has some other advantages, such as proxy signature key generation and updating using insecure channels. Our approach can also be applied to other ElGamal-like proxy signature schemes.

Acknowledgements:

References

1. R. Anderson and M. Kuhn, Tamper Resistance - A Cautionary Note, *Proc. Second USENIX Workshop on Electronic Commerce*, Oakland, CA, November 1996.
2. C. Boyd, Comment: New digital signature scheme based on discrete logarithm, *Electronics Letters*, Vol. 30, No. 6, 1994, pp 480-481.
3. D. Chaum, Blind Signatures for Untraceable Payments, *Advances in Cryptology: Proceedings of CRYPTO'82*, Plenum Press, 1983, pp. 199-203.
4. D. Chaum, Security Without Identification: Transaction Systems to Make Big Brother Obsolete, *Communications of the ACM*, v. 28, n. 10, 1985, pp 1030-1044.

5. D. Chaum, Zero-Knowledge Undeniable Signatures, *Proc. EUROCRYPT'90*, Springer Verlag, 1991, pp. 458-464.
6. D. Chaum, Designated Confirmer Signatures, *Proc. EUROCRYPT'94*, Springer Verlag, 1995, pp. 86-91.
7. J.L. Camenisch, J-M. Piveteau, and M.A. Stadler, Blind Signatures Based on the Discrete Logarithm Problem, *Proc. EUROCRYPT'94*, Springer Verlag, 1994, pp. 428-432.
8. D. Chaum and H. van Antwerpen, Undeniable Signatures, *Proc. CRYPTO'89*, Springer Verlag, 1990, pp. 212-216.
9. T. ElGamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Trans. on Information Theory*, Vol IT-31:4, 1985.
10. M. Franklin and M. Yung, The Blinding of Weak Signatures, *Proc. EUROCRYPT'94*, Springer Verlag, 1995.
11. P. Horster, H. Petersen, and M. Michels, Meta-ElGamal Signature Schemes, *Proc. 2nd Annual ACM Conference on Computer and Communications Security*, ACM Press, 1994, pp. 96-107.
12. P. Horster, M. Michels, and H. Petersen, Meta-Message Recovery and Meta-Blind signature schemes based on the discrete logarithm problem and their applications, *Proc. Asiacrypt'94, Lecture Notes in Computer Science 917*, Springer Verlag, 1995, pp. 224 - 237.
13. M. Mambo, K. Usuda and E. Okamoto, Proxy Signatures for Delegating Signing Operation, *Proc. 3rd ACM Conference on Computer and Communications Security*, 1996.
14. B. C. Neuman, Proxy-Based Authorization and Accounting for Distributed Systems, *Proc. 13th International Conference on Distributed Computing Systems*, May 1993, pp 283-291.
15. K. Nyberg and R.A. Rueppel, A New Signature Scheme Based on the DSA Giving Message Recovery, *Proc. 1st ACM Conference on Computer and Communications Security*, 1993.
16. K. Nyberg, Comment: New digital signature scheme based on discrete logarithm, *Electronics Letters*, Vol. 30, No. 6, 1994, pp 481.
17. K. Nyberg and R.A. Rueppel, Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem, *Proc. EUROCRYPT'94*, Springer-Verlag, pp. 182-193, 1995.
18. T. Okamoto, Designated Confirmer Signatures and Public-Key Encryption are Equivalent, *Proc. CRYPTO'94*, Springer Verlag, 1994, pp. 61-74.
19. S. Pohlig and M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, *IEEE Trans. on Information Theory*, 1978, IT-24, pp 106-110.
20. V. Varadharajan, P. Allen and S. Black, An Analysis of the Proxy Problem in Distributed Systems, *Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991, pp 255-275.
21. S.M. Yen and C.S. Laih, New Digital Signature Scheme Based on Discrete Logarithm, *Electronics Letters*, Vol. 29, No. 12, pp. 1120-1121, 1993.

A Nyberg-Rueppel Scheme

Nyberg-Rueppel scheme [15] is briefly described here. The system parameters consist of a prime p, a prime factor q of $p-1$, and an element $g \in Z_p^*$ of order q. The signer's

private key is a random element $x \in Z_q$, while the corresponding public key is $y = g^x \pmod{p}$. To sign a message m, which is an integer relatively prime to q, the signer selects $k \in Z_q$ at random and computes r and s as follows:

$$r = mg^k (\mathrm{mod}\, p)$$

$$s = xr + k (\mathrm{mod}\, q)$$

The pair (r, s) is the signature of the message m. To verify the validity of a signature, one checks that the following equality holds:

$$m = g^{-s} y^r r (\mathrm{mod}\, p).$$

B Blind Nyberg-Rueppel Scheme by Camenisch, Piveteau and Stadler

1. Alice selects $\tilde{k} \in Z_q$, computes $\tilde{r} = g^{\tilde{k}} (\mathrm{mod}\, p)$, and sends $\tilde{r}$ to Bob.
2. (a) Bob randomly selects $\alpha \in Z_q$ and $\beta \in Z_q^*$, computes $r = mg^{\alpha} \tilde{r}^{\beta} (\mathrm{mod}\, p)$ and $\tilde{m} = r\beta^{-1} (\mathrm{mod}\, q)$.
   (b) Bob checks whether $\tilde{m} \in Z_q^*$. If this is not the case, he goes back to step a). Otherwise, he sends $\tilde{m}$ to Alice.
3. Alice computes $\tilde{s} = \tilde{m}x + \tilde{k} (\mathrm{mod}\, q)$ and forwards $\tilde{s}$ to Bob.
4. Bob computes $s = \tilde{s}\beta + \alpha (\mathrm{mod}\, q)$.

The pair (r, s) is a Nyberg-Rueppel signature of the message m and the above protocol is a blind signature scheme.