

# A Provably Secure Proxy Signature Scheme with Multiple Threshold Values Based on Elliptic Curve

Zuhua Shao

Zhejiang University of Science and Technology  
No. 318, LiuHe Road, Hangzhou, Zhejiang, 310023, P. R. of China  
E-mail: [zhshao\\_98@yahoo.com](mailto:zhshao_98@yahoo.com)

**Abstract:** In this paper, we propose a threshold proxy signature scheme with multiple threshold values based on elliptic curve. The new scheme is simple, flexible and efficient since both the size of signature and verification computation do not depend on the number of actual proxy signers. Moreover, we show that the scheme is strongly unforgeable under the discrete logarithm assumption in the random oracle model.

**Keywords:** elliptic curve cryptosystems, threshold proxy signature scheme, multi-signature.

## I. Introduction

A proxy signature scheme [1] is a special type of signature scheme, which allows a designated person or a group of designated persons, called proxy signer, to sign messages on behalf of an original signer, a company or an organization. Proxy signature schemes have been shown to be useful in many applications. For example, a manager can delegate his secretaries to sign documents while he is on vacation.

A  $(t, n)$  threshold proxy signature scheme [2] is a variant of the proxy signature scheme. Any  $t$  or more proxy signers can cooperatively sign messages on behalf of the original signer, but  $t-1$  or fewer proxy signers cannot.

However, the ordinary  $(t, n)$  threshold proxy signature schemes available so far have only one predetermined threshold value, which maybe hinders them from solving some problems. For instance, for security reasons, a company may have a policy that checks of large value must be signed by more clerks rather than one person. For example, one clerk is permitted to sign checks of value less than \$1000 on behalf of the company, checks of value between \$1000 and \$10000 must be signed by at least  $t$  clerks, and checks of value larger than \$10000 must be signed by some more clerks. Therefore, such company needs to use a flexible proxy signature scheme to write checks of different value.

The proxy signature scheme with multiple threshold values can provide such company with a flexible tool to proxy sign messages according to different delegation policy.

## II. A proxy signature scheme with multiple threshold values based on elliptic curve

A proxy signature scheme comprises four phases.

### 2.1 System initialization phase

First, original signer  $P_0$  chooses an elliptic curve  $E(Z_p)$  defined over  $Z_p$  where  $p$  is a prime. Let  $Q \in E(Z_p)$  be a

base point of prime order  $q$ . The original signer  $P_0$  and each proxy signer  $P_i$  in the group  $PG = \{P_1, P_2, \dots, P_n\}$  has a private key  $a_i$  and the corresponding public key  $A_i = a_iQ$  certified by the Certificate Authority, ( $i = 0, 1, \dots, n$ ), respectively. The elliptic curve  $E(Z_p)$ , the primes  $p$  and  $q$ , and the point  $Q$  are made public. Moreover  $H(\cdot)$  and  $F(\cdot)$  are some suitably chosen collision-free hash functions.

### 2.2 Group secret key generation phase

The original signer  $P_0$  and  $n$  proxy signers negotiate a warrant  $w$  that contains the important proxy policy, including multiple threshold values  $t$ , the expiration time of the delegation of signing power, the identities and public keys of the original signer and the proxy signers.

Then the original signer  $P_0$  chooses a random  $r < q$  and computes  $R = rQ$ . Next he determines the value  $h = H(w, R)$  of collision-free hash function  $H$ . Then  $P_0$  computes the group secret (delegation) key  $d = a_0h + r \pmod q$ . Finally the original signer  $P_0$  sends  $(w, R, d)$  to each proxy signer  $P_i$  by a secure channel, for example, encrypting  $d$  with the public key  $A_i$  of the proxy signer  $P_i$ .

Each proxy signer would verify  $(w, R, d)$  by checking  $dQ = H(w, R)A_0 + R$ .

### 2.3 Proxy signature generation phase

Suppose that a message  $m$  needs to be proxy signed on behalf of the original signer  $P_0$ . According to the delegation policy stipulated in the warrant  $w$ , it should be signed at least  $t$  proxy signers. Let  $B$  be the subset of indices of  $t$  actual proxy signers. Without loss of generality, they are the proxy signers  $P_1, P_2, \dots, P_t$ .  $B = (b_1b_2 \dots b_n)_2$ , where if  $P_i$  engages in signing,  $b_i = 1$ , otherwise  $b_i = 0$ .

The first proxy signer  $P_1$  randomly chooses  $k_1$  in  $Z_q^*$ , computes  $K_1 = k_1Q$ , and sends  $K_1$  to the second proxy signer  $P_2$ .

The second proxy signer  $P_2$  randomly chooses  $k_2$  in  $Z_q^*$ , computes  $K_2 = k_2Q + K_1$ , and sends  $K_2$  to the third proxy signer  $P_3$ .

.....

The last proxy signer  $P_t$  randomly chooses  $k_t$  in  $Z_q^*$ , computes  $K_t = k_tQ + K_{t-1}$  and sends  $K = K_t$  to the first proxy signer  $P_1$ .

The first proxy signer  $P_1$  computes  $f_0 = F(m, K, h, B, 0)$ ,  $f_1 = F(m, K, h, B, 1)$  and  $s_1 = k_1 + f_0d + f_1a_1 \pmod q$  and sends them together with  $K$  to  $P_2$ .

The second proxy signer  $P_2$  can verify the validity of  $s_1$  by checking the following equation:

$$s_1Q = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + K_1.$$

Then,  $P_2$  computes  $f_2 = F(m, K, h, B, 2)$  and  $s_2 = s_1 + k_2 + f_2a_2 \pmod{q}$  and sends them together with  $K$  to  $P_3$ .

The third proxy signer  $P_3$  can verify the validity of  $s_2$  by checking the following equation:

$$s_2Q = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + F(m, K, h, B, 2)A_2 + K_2.$$

.....

The last proxy signer  $P_t$  can verify the validity of  $s_{t-1}$  by checking the following equation:

$$s_{t-1}Q = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + \dots + F(m, K, h, B, t-1)A_{t-1} + K_{t-1},$$

where  $h = H(w, R)$ . Then,  $P_t$  computes  $f_t = F(m, K, h, B, t)$  and  $s = s_{t-1} + k_t + f_t a_t \pmod{q}$  and broadcasts  $(m, w, R, B, s, K)$  as the proxy signature of the message  $m$  with the warrant  $w$ .

#### 2.4 Verification of the proxy signature

To verify the threshold proxy signature  $(m, w, R, B, s, K)$ , the verifier checks the warrant  $w$ . If the validity has expired or the message needs to be signed by more than  $t$  proxy signers, the proxy signature is invalid. Otherwise the verifier checks

$$sQ = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + \dots + F(m, K, h, B, t)A_t + K,$$

where  $h = H(w, R)$ . It is easy to see that if  $(m, w, R, B, s, K)$  is generated by the proxy signature generation algorithm, it would pass the verification algorithm.

### III. Security model and security proof

In fact, this threshold proxy signature is a multi-signature of 2 messages signed by  $t+1$  signers, which is a generalization of the multi-signature in the plain public key model due to Bellare and Neven [3]. They proposed a security model assuming that there is a single honest signer. The adversary has effectively corrupted all other signers. It can choose their public keys as it likes, even as a function of the public key of the honest signer, and can interact arbitrarily with the honest signer in any number of concurrent instances before outputting its forgery. In such security model, they showed that the multi-signature scheme is proven secure in the random oracle model under the standard DL assumption.

#### 3.1 Security model

Existential unforgeability against adaptive chosen message attacks (EUF-CMA) [4] is the well-accepted security model for signature schemes, where the adversary is allowed to ask the challenger to sign any message of its choice adaptively, i.e. he can adapt its queries according to previous answers. Finally, the adversary could not provide a new message-signature pair with a non-negligible advantage.

The underlying signature of the proposed scheme, the Schnorr signature, is not deterministic. The signer may generate several signatures corresponding to a given message. We adopt a stronger security model, strong unforgeability, where the adversary is allowed to ask for

signatures of the same message many times, and he would obtain some useful information from each new answer. The adversary is required to forge a new signature on a previously signed message. This model gives the adversary more powers and more chances for success.

In the proposed threshold proxy signature scheme, besides ordinary adversaries of signatures, there are two types of adversaries with more powers than ordinary adversaries. Type I adversaries are a group of undelegated proxy signers, who want to forge proxy signatures of an innocent original signer by using public keys of their choice except for that of the innocent original signer. The adversaries are allowed to ask for the group secret key associated with any warrant  $w$  of their choice. We refer to such queries as delegation queries. Type II adversaries are a group of proxy signers and an original signer, who want to forge proxy signatures relative to an innocent proxy signer by using public keys of their choice except for that of the innocent proxy signer.

#### Type I attack

We say that a threshold proxy signature scheme is strongly unforgeable against the Type I attack if no polynomial bounded Type I adversaries have a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter  $1^k$  and runs the key-generation algorithm. It gives the Type I adversaries the resulting system parameters and a random public key  $A_0$  of the original signer  $P_0$ .

Queries: The Type I adversaries issue queries  $q_1, \dots, q_m$  adaptively where query  $q_i$  is one of:

- Delegation query  $\langle w \rangle$ , where  $w$  includes the public keys  $A_1, \dots, A_n$  chosen by the Type I adversaries besides the given public key  $A_0$ .

- Threshold proxy sign query  $\langle w, m, B \rangle$ .

Output: Finally, the Type I adversaries output a new proxy signature  $(m, w, R, B, s, K)$  for a message  $m$  with respect to a warrant  $w$  composed by the adversaries, which includes proxy signer's public keys  $A_1, \dots, A_t$  chosen by the adversaries besides the challenged public key  $A_0$ .

The Type I adversaries win the game if the output signature  $(m, w, R, B, s, K)$  is nontrivial, i.e. it is not an answer of a threshold proxy sign query for the message  $m$  and the warrant  $w$ , and  $R$  is not an answer of a delegation query  $\langle w \rangle$ .

The probability is over the random bits used by the challenger and the Type I adversaries.

#### Type II attack

We say that a threshold proxy signature scheme is strongly unforgeable against the Type II attack if no polynomial bounded Type II adversaries have a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter  $1^k$  and runs the key-generation algorithm. It gives the Type II

adversaries the resulting system parameters and a random public key  $A_s$  of a proxy signer  $P_s$ .

Queries: The Type II adversaries issue queries  $q_1, \dots, q_m$  adaptively, where query  $q_i$  is one of:

- Threshold sign query  $\langle w, m, B \rangle$ , where  $w_i$  includes the public keys  $A_0, \dots, A_n$  chosen by the Type II adversaries besides the given public key  $A_s$ .

Output: Finally, the Type II adversaries outputs a new threshold proxy signature  $(m, w, R, B, s, K)$  with respect to a warrant  $w$  composed by the adversaries, which includes the public keys  $A_0, \dots, A_t$  chosen by the Type II adversaries besides the challenged public key  $A_s$ .

The Type II adversaries win the game if the output signature  $(m, w, R, B, s, K)$  is nontrivial, i.e. it is not an answer a threshold proxy sign query for the message  $m$  and the warrant  $w$ .

The probability is over the random bits used by the challenger and the Type II adversaries.

### 3.2 Security proof of the threshold proxy signature scheme

We have the following theorem about the security of the threshold proxy signature scheme.

**Theorem.** Suppose that there are a group of type I adversaries  $A$  that have advantage  $\varepsilon$  against the threshold proxy signature scheme and  $A$  run in time at most  $tt$ . Suppose that  $A$  make at most  $q_H, q_F$  queries to the hash functions  $H$  and  $F$  respectively, at most  $q_D$  queries to the delegation oracle, and at most  $q_{TS}$  queries to the threshold proxy sign oracle. Then there is an algorithm  $B$  that  $(tt', \varepsilon')$ -breaks DL, where:

$$\varepsilon \leq q_H q_F (2^{13} \varepsilon')^{1/7} + 1/q + q_{TS}(t+1)(q_F + (t+1)q_{TS})/q + q_D(q_H + q_D)/q \quad (1)$$

$$tt \approx tt'/4 - ((t+3)q_{TS} + 2q_D)C_{sm}. \quad (2)$$

Here  $C_{sm}$  denotes the computation of a scalar multiple in the elliptic curve.

Proof: We show how to construct a DL algorithm  $B$  that uses  $A$  as a computer program to gain an advantage  $\varepsilon'$  for a DL problem with running time  $tt'$ . The challenger takes a security parameter  $1^k$  and runs the key-generation algorithm to generate the system parameters and  $A_0$ . Its goal is to output  $a_0 = \log_Q A_0$  with  $A_0 = a_0 Q$ .

Algorithm  $B$  simulates the challenger and interacts with the Type I adversaries  $A$  in the following attack games:

Algorithm  $B$  gives the adversaries  $A$  the resulting parameters and  $A_0$  as the public key of the original signer. At any time, the adversaries  $A$  can query hash oracles  $H$  or  $F$ . To response to these queries,  $B$  maintains two lists of tuples for the hash oracles  $H$  and  $F$ , respectively. We refer to these lists as  $H$ -list and  $F$ -list. The contents of the two lists are "dynamic" during the attack games. Namely, when the games start, they are initially empty, but at the end of the games, they record all pairs of queries/answers.

*Answering H-oracle Queries.* For a new query  $\langle w, R \rangle$ , algorithm  $B$  picks a random  $h$  in  $Z_q^*$ , and responds with  $h = H(w, R)$  and adds the tuple  $\langle \langle w, R \rangle, h \rangle$  to the

$H$ -list.

*Answering F-oracle Queries.* For a new query  $\langle m, K, h, B, i \rangle$ ,  $B$  checks if  $h$  is in the  $H$ -list and generates a random  $e_i \in Z_q^*$  and responds with  $e_i = F(m, K, h, B, i)$  and adds the tuple  $\langle \langle m, K, h, B, i \rangle, e_i \rangle$  to the  $F$ -list.

Obviously, in two ways,  $h$  and  $e_i$  are uniform in  $Z_q^*$ , and they are independent of  $A$ 's current view as required.

*Answering delegation queries.* For a new delegation oracle  $\langle w \rangle$ ,  $B$  checks if  $w$  is a valid warrant. Then  $B$  generates two random integers  $h$  and  $d \in Z_q^*$  and computes  $R = dQ - hA_0$ , where  $A_0$  is the public key of the original signer in  $w$ .

If there exists a tuple  $\langle \langle w, R \rangle, h' \rangle$  in the  $H$ -list with  $h \neq h'$ ,  $B$  aborts and restarts simulation (the probability of this unfortunate coincidence is at most  $(q_H + q_D)/q$ ).

$B$  answers with  $(w, R, d)$ , and adds the tuple  $\langle \langle w, R \rangle, h \rangle$  to the  $H$ -list.

*Answering threshold proxy sign queries.* For a new threshold proxy signature  $\langle w, m, B \rangle$ ,  $B$  checks if  $w$  is a valid warrant. Then,  $B$  picks a random  $R \in rQ$  where  $r \in Z_q^*$  and runs the above algorithm for responding to  $H$ -queries to obtain  $h = H(w, R)$ .

$B$  chooses at random  $s$  and  $e_0, e_1, \dots, e_t \in Z_q^*$ , and computes  $K = sQ - e_0(hA_0 + R) - e_1A_1 - \dots - e_tA_t$ , where  $A_0$  is the public key of the original signer and  $A_1, \dots, A_t$  are the public keys of a group of proxy signers indexed in  $B$ .

If there exists a tuple  $\langle \langle m, K, h, B, i \rangle, e_i' \rangle$  in the  $F$ -list with  $e_i \neq e_i'$ , ( $i = 0, 1, \dots, t$ ),  $B$  reports failure and terminates. (The probability of this unfortunate coincidence is at most  $(t+1)(q_F + (t+1)q_{TS})/q$ ).

Otherwise,  $B$  responds with  $(m, w, R, B, s, K)$  to the adversaries  $A$  and adds  $\langle \langle m, K, h, B, i \rangle, e_i \rangle$  ( $i = 0, 1, \dots, t$ ) to the  $F$ -list.

Obviously, the outputs of the simulated oracles are indistinguishable from those in the real attacks.

Finally, the adversaries  $A$  return a new threshold proxy signature  $(m, w, R, B, s, K)$  such that

$$sQ = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + \dots + F(m, K, h, B, t)A_t + K,$$

where  $h = H(w, R)$ .

If the adversaries  $A$  have not queried  $F(m, K, h, B, i)$  ( $i = 0, 1, \dots, t$ ) or  $H(w, R)$ , the probability of

$$\Pr[sQ = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + \dots + F(m, K, h, B, t)A_t + K] \leq 1/q,$$

since both the responses to hash functions are picked randomly.

Hence, with the probability

$$(1 - 1/q)(\varepsilon - q_{TS}(t+1)(q_F + (t+1)q_{TS})/q - q_D(q_H + q_D)/q) \geq (\varepsilon - 1/q - q_{TS}(t+1)(q_F + (t+1)q_{TS})/q - q_D(q_H + q_D)/q)$$

the Type I adversaries  $A$  return a new threshold proxy signature such that

$$sQ = F(m, K, h, B, 0)(H(w, R)A_0 + R) + F(m, K, h, B, 1)A_1 + \dots + F(m, K, h, B, t)A_t + K,$$

where  $h = H(w, R)$ , and the adversaries  $A$  have queried  $F(m, K, h, B, i)$  ( $i = 0, 1, \dots, t$ ) and  $H(w, R)$ .

Note that the public keys  $A_1, A_2, \dots, A_t$  are chosen by the adversaries A and the public key  $A_0$  is chosen by the challenger.

We try to use oracle replay techniques of Pointcheval and Stern [5] to solve this DL problem, finding  $a_0$  such that  $A_0 = a_0Q$ . B uses four copies of the Type I adversaries A.

In the attack games, the Type I adversaries A should query  $H(w, R)$ . We first guess a fixed index  $1 \leq k \leq q_H$  and hope that  $\langle w, R \rangle_k$  happens to be one for which A ask for  $h = H(w, R)$  query. Then we guess a fixed index  $1 \leq j \leq q_F$  and hope that  $e_0 = F(m, K, h, B, 0)_j$  happens to be one for which A forge a threshold proxy signature. A must first ask for  $H(w, R)$  before for  $F(m, K, h, B, i)$ .

Suppose that we make two good guesses by chance, denoted by the event GoodGuess. The probability of the event GoodGuess is

$$\Pr[\text{GoodGuess}] = 1/(q_H q_F).$$

Hence, with the probability

$$\epsilon'' \geq (\epsilon - 1/q - q_{TS}(t+1)(q_F + (t+1)q_{TS})/q - q_D(q_H + q_D)/q)/(q_H q_F)$$

the adversaries A generate a new threshold proxy signature.

B gives the same system parameters, the public key  $A_0$  and same sequence of random bits to the four copies of the adversaries A, and responds with the same random answers to their queries for oracles until they at the same time ask the  $H$ -oracle query for  $\langle w, R \rangle$ . This is the first forking point. At that point, B gives two independent random answers  $h_1$  and  $h_2$  to the hash queries  $H(w, R)$ , the first two, gives  $h_1$ , and the last two, gives  $h_2$ .

Then B gives the first two copies of the adversaries A same sequence of random bits, and the same random answers to their oracle queries until they both ask for  $F(m, K, h, B, 0)$ . This is the second forking point. At that point, B gives two independent random answers  $e_{011}$  and  $e_{012}$  to the hash queries  $F(m, K, h, B, 0)$  in the first two runs. Meanwhile B gives the same random answers to the hash queries  $F(m, K, h, B, i)$  ( $i = 1, \dots, t$ ) in the two runs.

Similarly, B gives two independent random answers  $e_{021}$  and  $e_{022}$  to the hash queries  $F(m', K', h, B', 0)$  (the third forking point) and gives the same random answers to the hash queries  $F(m', K', h, B', i)$ , ( $i = 0, 1, \dots, t$ ), in the last two runs. Thus, we would obtain four threshold proxy signatures, satisfying the following equations:

$$\begin{aligned} S_{11}Q &= e_{011}(h_1A_0 + R) + e_{11}A_1 + \dots + e_{t1}A_t + K \\ S_{12}Q &= e_{012}(h_1A_0 + R) + e_{11}A_1 + \dots + e_{t1}A_t + K \\ S_{21}Q &= e_{021}(h_2A_0 + R) + e_{12}A_1 + \dots + e_{t2}A_t + K' \\ S_{22}Q &= e_{022}(h_2A_0 + R) + e_{12}A_1 + \dots + e_{t2}A_t + K'. \end{aligned}$$

From these equations, we have

$$\begin{aligned} (S_{11} - S_{12})/(e_{011} - e_{012})Q &= (h_1A_0 + R) \\ (S_{21} - S_{22})/(e_{021} - e_{022})Q &= (h_2A_0 + R). \end{aligned}$$

Then we can derive  $a_0$  such that  $A_0 = a_0Q$ , since  $h_1$  and  $h_2$  are different.

We continue to use the ‘‘splitting lemma’’ [5] to compute the probability that A work as hoped. Let  $X$  be

the set of possible sequences of random bits and random function values that take the adversaries up to the first forking point where A ask for  $H(w, R)$ ; let  $Y$  be the set of possible sequences of random bits and random function values from the first forking point to the second forking point, where A ask for  $F(m, K, h, B, 0)$ ; let  $Z$  be the set of possible sequences of random bits and random function values from the second forking point. By assumption, for any  $x \in X, y \in Y, z \in Z$ , the probability that A, supplied the sequences of random bits and random values  $(x||y||z)$ , generate a proxy signature is  $\epsilon''$ .

Suppose that the sequences of random bits and random function values supplied up to the first forking point in the simulations is  $a$ . By ‘‘splitting lemma’’,  $\Pr\{a \in \text{‘‘good’’ subset } \Omega\} \geq \epsilon''/2$ , and whenever  $a \in \Omega, y \in Y, z \in Z$ , the probability that A, supplied the sequences of random bits and random values  $(a||y||z)$ , produce a forgery is at least  $\epsilon''/2$ .

Suppose that the sequences of random bits and random function values, supplied from the first forking point up to the second forking point in the simulations, is  $b$ . Thus,  $\Pr\{b \in \text{‘‘good’’ subset } \Omega'\} \geq \epsilon''/4$ , and whenever  $a \in \Omega, b \in \Omega', z \in Z$ , the probability that A, supplied the sequences of random bits and random values  $(a||b||z)$ , produce a forgery is at least  $\epsilon''/4$ .

By the same reason, we can compute the same probability for the other case.

Hence the probability that B solves the discrete logarithm through the four simulations is

$$\epsilon' \geq (\epsilon'')^7/2^{13} \geq ((\epsilon - 1/q - q_{TS}(t+1)(q_F + (t+1)q_{TS})/q - q_D(q_H + q_D)/q)/(q_H q_F))^7/2^{13}.$$

The time required to run one simulation is  $tt + ((t+3)q_{TS} + 2q_D)C_{sm}$ . The time required to solve the discrete logarithm  $\log_Q A_0$  is

$$t' \leq 4(tt + ((t+3)q_{TS} + 2q_D)C_{sm}). \quad \text{Q.E.D.}$$

By a similar way, we can show that the proposed threshold proxy signature scheme is secure against the Type II adversaries. Therefore, we obtain the Theorem.

## REFERENCES

- [1] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures: Delegation of the power to sign messages, *IEICE Trans. Fundam.*, E79-A, (9), pp.1338-1354, 1996.
- [2] H.-M. Sun. An efficient nonrepudiable threshold proxy signature scheme with known signers. *Computer Communications*, 22(8): 717-722, 1999.
- [3] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a generalized forking lemma, *CCS 2006*, pp.390-399, ACM 2006.
- [4] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks, *SIAM Journal on Computing*, 17(2), pp.281-308, 1988.
- [5] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures, *Journal of Cryptology*, 13(3), pp. 361-396, 2000.