Practical and Secure E-Mail System (PractiSES)

Albert Levi and Mahmut Özcan

Sabanci University
Faculty of Engineering and Natural Sciences, Orhanli, Tuzla, TR-34956 Istanbul, Turkey
levi@sabanciuniv.edu mozcan@sampas.com.tr

Abstract. In this paper, a practical and secure e-mail system (called "PractiSES") that performs public key distribution and management in a unique way is proposed. PractiSES is a network of several domains. At the domain level, a designated PractiSES server, which is trusted by all users of that domain, distributes the public keys. If a user needs another user's public key at a different domain, then inter-domain communication is carried out. PractiSES clients manage their public keys and obtain public keys of other users by using unique, secure and user-transparent protocols. PractiSES clients can exchange e-mails in encrypted and/or signed fashion. Since on-demand fetching of public keys is aimed in PractiSES, use of certificates is limited for inter-domain communications only; no certificates are used within a domain. Our simulations show that a state-of-the-art PC would be sufficient to serve as PractiSES server of a medium-size organization.

1 Introduction

E-mail is one of the most commonly used communication mechanisms. Most of the recipients and senders desire secure e-mail exchange. Senders want to make sure that the recipient is really the intended recipient, and the message arrives to the recipient confidentially. On the other hand, recipients want to make sure that the sender is the entity who it claims to be, and the arrived message has not been maliciously modified and examined during transmission. These requirements can be satisfied by the e-mail applications that use public key cryptosystem (PKC) as the security base, such as S/MIME [1] and PGP [2]. The main handicap behind the deployment of applications that use PKC is the problem of public key distribution with a legitimate binding with its owner. Moreover, public key management features, such as update, delete operations must be performed in a secure way.

S/MIME is a standard mechanism that provides secure Internet message exchange between parties. S/MIME applications use digital certificates for public key distribution. Certificates [3] are digital documents that are used as bindings between users' identities and their public keys. Use of certificates may be inconvenient for several reasons. Certificates can be downloaded from certificate repositories and e-mail addresses in certificates may be collected by e-mail address collectors for mass mailing. Privacy sensitive people criticize this situation. Certificates that are issued by well-known Certification Authorities (CAs) are not free. Certificates are used in offline manner, so revocation of them is a troublesome process.

There is no common trusted third party in PGP [2]. Rather, every user can certify another user. Therefore, a message from a user, who is certified by another user not known to the receiver, may cause the receiver to hesitate. PGP has a network of public key servers, but key authenticity decisions are eventually given by the user itself. Thus, PGP key management and distribution mechanisms assume knowledgeable users.

In this paper, under the light of PGP and S/MIME experience, we propose a new secure e-mail system, *Practical and Secure E-Mail System* (*PractiSES*), that is somehow similar to a two-tiered Public Key Infrastructure (PKI) with online key servers. At the top level of PractiSES system, there is a Certificate Authority (CA), called *PractiSES CA*, to provide public key certificates for online PractiSES domain servers. Public keys of users of a specific domain are stored, distributed and managed in a centralized manner by employing a domain server and a public key storage of that domain. No certificates are used for such intra-domain public key distribution. Users belong to different domains exchange their public keys via inter-domain communication in which domain server certificates are used.

The rationale behind the design of PractiSES is explained in Section 2. Design of PractiSES and its protocols are given in Section 3. Implementation and performance details are explained in Section 4. Finally, conclusions and discussions on PractiSES are given in Section 5.

2 The Rationale Behind PractiSES

While designing PractiSES as a secure and practical e-mail system, we have considered two facts about e-mail systems.

First, we have realized that an e-mail security system is useless unless both parties use it. If a recipient does not use a secure e-mail client, sender's signature over a message is worthless. Both PGP and S/MIME suffer from this fact. A good system should be aware of the recipient's capabilities while sending a secure e-mail.

Second, we have realized that neither PGP nor S/MIME achieved a critical mass in order to be considered as default e-mail security mechanisms. Certificate requirement is the shortcoming of S/MIME. PGP's problem is at its complexity. So we need a user friendly and simple system that does not require end user certificates and eliminates the related problems.

These two observations led us to a centralized approach for e-mail security. In PractiSES, a centralized server could store the public keys and distribute them on-demand basis to the other users in an online manner. In this way, we eliminate all certificate related problems. Since this server is a trusted one, users do not bother with complex trust decision systems. Another attractiveness of such a centralized approach for closed groups is that it is plausible to assume the system knows its potential users. In this way, potential users' e-mail addresses and semi-secret information, such as mother's maiden name and SSN, can be stored in the server's database. This information is later used for authentication during the initial registration.

The above-described centralized approach solves the problem only within a domain of users. It is obvious that all potential users cannot exist within a single

domain, so inter-domain end user public-key transfers are needed. We prefer to use a certificate-based mechanism, which will be detailed in the coming sections, for inter-domain secure communication. One may argue that it is contradictory to use certificates for domains while the design decision was not to use end user certificates. However, we believe that since the number of domains is not too much (especially as compared to the number of end users), the managerial problems of using certificates for domains become tractable. Moreover, the domain certificates are used transparent to the end users. When a domain certificate is revoked, which is quite unlikely, only other domain servers should handle this, not the end users. In such a case, PractiSES CA informs all the domain servers.

3 Design of Practical and Secure E-mail System (PractiSES)

From a high-level point of view, PractiSES is a network of several *PractiSES domains* and a *PractiSES Certificate Authority* (*CA*). Every PractiSES domain has a server, called *PractiSES Server*, which holds its key pair and PractiSES CA's self-signed root certificate, which may come with the server software package. Moreover, each server, consequently each domain, has a key pair and public key is available to other domains via a domain certificate pre-issued by PractiSES CA. Thus, there is a certification relationship between a PractiSES domain and CA. The trust between servers of different domains is established by using those certificates. Whenever an end user public key is to be exchanged between two domains, domain servers authenticate themselves via a protocol, key obtainment protocol, in which certificates are involved. Figure 1 shows a high-level overview of an example PractiSES system with three domains (X, Y and Z).

3.1 PractiSES Domain Architecture and Client Module

A PractiSES Domain has a centralized architecture. A designated domain server, *PractiSES Server*, acts as a key distribution center for the whole domain users. Most of those users are expected to be affiliated with the same institution, but this is not a must; any user can be served in the domain of a PractiSES Server. Public keys are eventually stored in *Public Key Storage*, but that storage initially does not contain public keys but contains values of ID, name, last name, e-mail address and semi-secret information (e.g. mother's maiden name), etc. of the potential users. It is assumed that such information exists in organization's records and can easily be conveyed to the public key storage. This potential user information will be used for authentication during the initialization protocol to upload the public keys of the users.

Moreover, each client obtains the public key of its domain server. This is necessary to authenticate the messages coming from the server and to send encrypted messages to it. This public key is stored in self-signed format and downloaded from a designated web or ftp site with a manual key digest crosscheck to make sure about the authentication of the key. Should a domain key is revoked, domain users are informed with a signed e-mail from a domain server so that they download new public key from the same site.

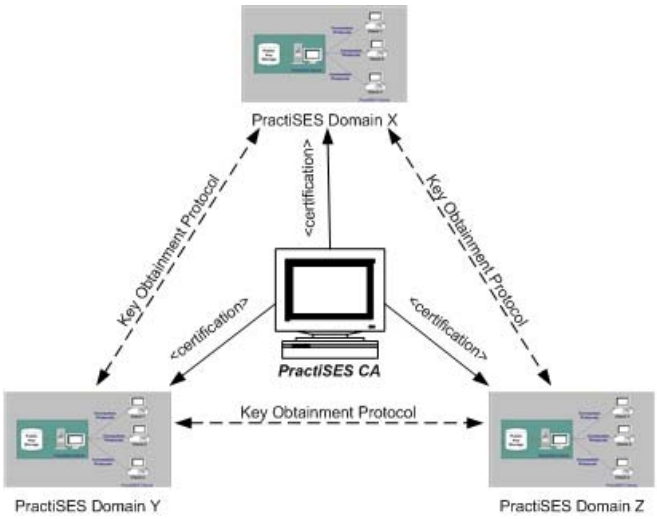

Fig. 1. Practical and Secure Email System (PractiSES)

As can be seen from the above discussion, PractiSES protocols assume pre-distribution of server public and CA public keys in an offline manner. This is a general problem for all security applications and solution generally requires sacrificing security to some extent. For example, as explained in [10], the root CA certificates for the well-known SSL (Secure Socket Layer) protocol come with browsers and installed automatically without the consent of the user. In that respect, PractiSES follows the commonsense.

The Client Module (CM) is an application, which is used as an e-mail client with additional security options. It has two functionalities with different security features. One functionality is for key management. The other functionality is secure e-mail transfer. Key management is implemented as a set of secure protocols that will be examined in Section 3.2 in detail. They are used for key pair generation, initialization and public key upload, key obtainment, key update, and key removal purposes. Secure e-mail exchange is implemented in the CM as the capabilities of sending and receiving signed and/or encrypted e-mails on top of normal e-mail client operations.

3.2 Connection Protocols

Connection protocols are designed to accomplish different key management and distribution operations in a secure way. They are listed below:

1. Initialization and Public Key Settlement Protocol (*InitKeySet*)
2. Public Key Obtainment Protocol (*KeyObt*)
3. Public Key Update Protocol (*KeyUpdate*)
4. Public Key Removal Protocol (*KeyRem*)
5. Unsigned Public Key Removal Protocol (*USKeyRem*)
6. Unsigned Public Key Update Protocol (*USKeyUpdate*)

All of these protocols run within a domain except the key obtainment protocol. In *KeyObt* protocol, a domain server may need to talk to another domain server if public key of a user at a different domain is needed.

Initialization and Public Key Settlement Protocol (InitKeySet). This protocol is designed for users to upload their public keys to the PractiSES domain server (public key storage) for the first time. In order to authenticate the clients, server uses clients' private information that already exists in the public key storage, such as ID, shared-secret, birthday and some other identity information. The sequence diagram of *InitKeySet* protocol is presented in Figure 2.

1. The end user introduces himself/herself to the server by providing his/her ID and e-mail address.
2. The server retrieves the user's information stored in the public key storage. Then it matches the user ID and e-mail address received from the previous step with the ones in the public key storage. If they match, then the server asks the user for shared semi-secret information. Server signs the questions before sending them out.
3. User verifies server's signature over the questions. If valid, then he/she encrypts the answers using a randomly generated secret key and encrypts the same secret key using server's public key. The user sends all these encrypted data to the server.
4. Server decrypts the secret key using its own private key. Then it decrypts the user's answers using the secret key. The answers are compared to the corresponding information stored in the server's database. If they match, then the server sends an e-mail that contains a server-given Message Authentication Code (MAC) [4] password encrypted with the same secret key. Otherwise protocol stops.
5. The user accesses his/her inbox and retrieves the e-mail that contains the encrypted MAC password. The, he/she decrypts the MAC password using the secret key that he/she already knows from previous steps. This password is used to provide integrity and authenticity for the message that contains user's public key.
6. Server checks the MAC within the message that contains the user's public key. If verified, then public key is deemed legitimate and stored in the public key storage. Server sends a confirmation message about successful/unsuccessful key upload.

Security of the protocol. An attacker who eavesdrops on the communication link between the client and the server cannot upload a fake public key since such an upload requires knowing the MAC password and consequently the secret key, which is generated by the client and encrypted using server's public key. The attacker cannot obtain those secrets.

Authentication of the client to the PractiSES server is a challenging issue since we do not assume that there is pre-shared full secret (such as a PIN number) between them due to practical difficulty of distributing them. However, if such a secret is pre-distributed, then the server always asks that value in step 2 of the protocol, so that authentication can be 100% assured. In the protocol, we assume semi-secrets shared between client and the server. An attacker who knows these secrets may try to

impersonate a client starting with the beginning of the protocol. However, the attacker cannot fool the server to use an e-mail address for the client other than the one stored in its records. Thus, the attacker should have a continuous access to the client's actual inbox not only to obtain the MAC password to upload the bogus public key, but also to utilize the attack after the fake upload. We believe that such an environment, in which the attacker knows the semi-secrets and has the access to the client's inbox that is separately secured with traditional username/password mechanism, is not so likely in practice.

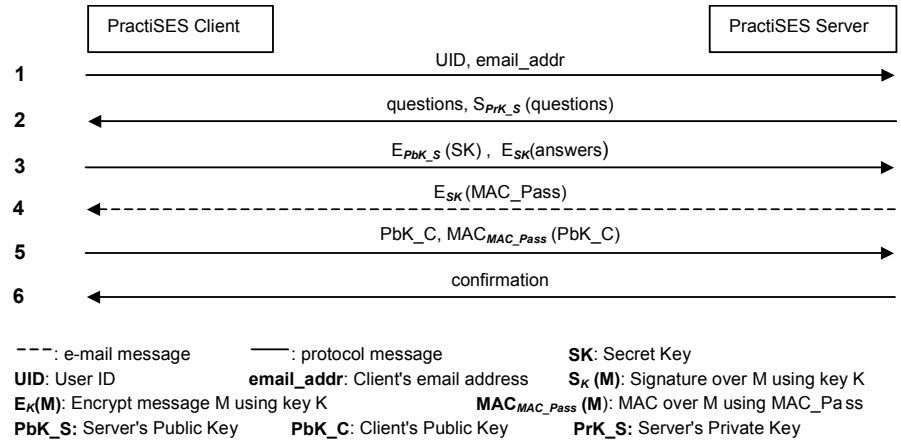

Fig. 2. Sequence Diagram of InitKeySet Protocol

Public Key Obtainment Protocol (KeyObt). *KeyObt* protocol is designed to obtain another user's public key from the public key storage with the server's signature on it. If the user of the public key requested belongs to the same domain with the requester, then the request is responded by the server of the home domain. If a public key owned by a user in a different domain is needed, then the home domain server needs to contact server of target user's domain. This protocol is presented in Figure 3.

1. Requesting end user sends the e-mail address of the target user (the user whose public key is being sought), the common name of the target user's domain and a randomly generated nonce value.

Then the home server checks the domain's common name. If it is the same with home domain, then home server retrieves target's public key from its public key storage and continues with step 4. If the common name is foreign, then home server contacts with the server of foreign domain and continues with step 2.

2. Home server sends target user's e-mail address, common name of target user's domain, and newly generated nonce value of home server to the foreign server.

3. First foreign server retrieves target user's public key from its public key storage. Then, it responses to the home server with a message, which comprises from data and signature parts. Data part includes the values of the name, last name, e-mail address, and public key of the target user, the received nonce value of home server, and the certificate of foreign server issued by PractiSES CA. Here, certificate is a digitally signed document that provides a binding between foreign server's common name and its public key. The common name and public key of foreign server, validity period of the document and other fields (issuer name, serial number, etc.) constitute the certificate of foreign server. Foreign server signs the data part with its private key. In this way, home server makes sure about the legitimacy of the public key and the other information it received. Home server takes the response of foreign server, checks the correctness and validity of foreign server's certificate.
4. At this step, home server constructs the response message, which comprises from the name, last name, e-mail address, public key of the target user and the client's nonce value. Then it sends this message to the client with a signature generated using home server's private key. Client gets the signed message and verifies it using home server's public key. The public key of target user is ready for e-mail.

Security of the protocol. In the protocol, nonce values are used to assure the freshness of the response of responders. Suppose an attacker records a server response for a key. Also assume that the same key is deleted at a later time. If the attacker replays the recorded server response for that key back to a requester after the deletion time, it can easily reintroduce the invalid key as valid. Using a nonce value in the protocol prevents such replay attacks since the requester would not accept a response that includes a nonce value it did not generate recently.

Since the public keys are not secrets, clients need not to be authenticated in order to request other users' public keys. However, the server should authenticate its response, which is implemented by the digital signature in steps 3.2 and 4. An attacker cannot forge that signature since it does not know of the private keys of the servers. Therefore, the attacker cannot present a fake public key as valid.

Other Protocols. The details of other protocols (*KeyUpdate*, *KeyRem, USKeyRem* and *USKeyUpdate*) are skipped for the sake of brevity. *KeyRem* and *KeyUpdate* are designed for the clients to remove and update their public keys at the PractiSES server, respectively. The security of the update/remove requests sent from client to server is provided by signing them using client's current private key. *USKeyRem* and *USKeyUpdate* are designed also for removing and updating public keys. The difference of these protocols from *KeyRem* and *KeyUpdate* is that *USKeyRem* and *USKeyUpdate* use Message Authentication Code (MAC) instead of digital signatures for integrity and authentication of the request. As in the *InitKeySet* protocol, the MAC password is sent out in an e-mail after a semi-secret-check type of authentication. In this way, the users can remove and update their keys securely even if they cannot authenticate themselves by using their private keys due to loss or compromise.

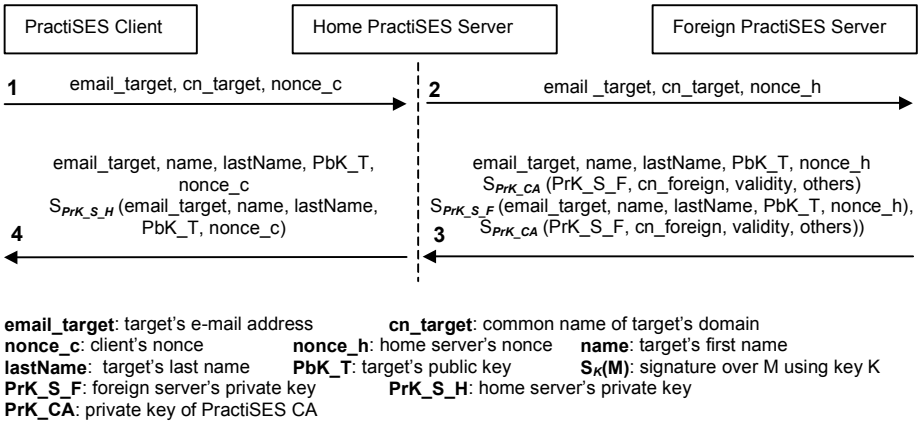

Fig. 3. Sequence Diagram of KeyObt Protocol

4 Performance and Implementation Issues

Connection protocols and secure e-mail client are implemented in Java. The protocols are leveraged on the TCP/IP protocol stack. PractiSES uses the Java Cryptographic Environment (JCE 1.2.2) for cryptographic primitives such as (i) 2048-bit RSA [5, 7] algorithm for public key encryption/decryption and together with SHA-1 [8] for digital signatures, (ii) Triple-DES (3-DES) [6] for conventional encryption and decryption, and (iii) Hash-Based MAC (HMAC) [9] function to provide message integrity and authentication.

In our simulations, we evaluate the performance PractiSES by analyzing the throughput of the PractiSES server and end-to-end latency. As the server, we use an ordinary PC with 2.5 GHz P4 processor, and as the client a PC with 1GHz P3 processor. Figure 4 shows that the server throughput decreases due to increased queuing delay as the number of key obtainment requests increase. For example, when the average key obtainment request is 35 requests/sec, the average server throughput is 24.35, that corresponds to $1000/24.35 \approx 41$ msec of average total server residency time (including waiting and processing times) per request. Since the server processing time is about 20 msec/request under light loads, and extra 21 msec would be considered as a good trade-off for having an average of 35 requests/sec that corresponds to about 1 million key requests per 8-hour period which is a reasonable daily load for a medium size organization.

The latency as seen by the client is another performance indicator. This latency includes key obtainment protocol run, cryptographic processing over the message and networking delays. The most important factor is, as in the previous analysis, is the load on the server. Figure 5 shows how the latency of message verification increases as the load on server gets larger. For the average load of 35 requests/second, which we used in our example for the previous analysis, the end-to-end latency is about 320 msec; in other words, PractiSES, on the average, costs an extra 320 msec for message verification that is a quite imperceptible delay in e-mail reading.

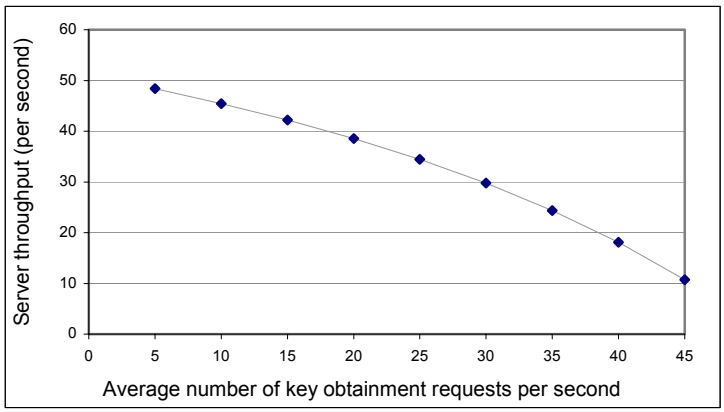

Fig. 4. The change of server throughput in terms of requests processed per second as average number of key requests changes

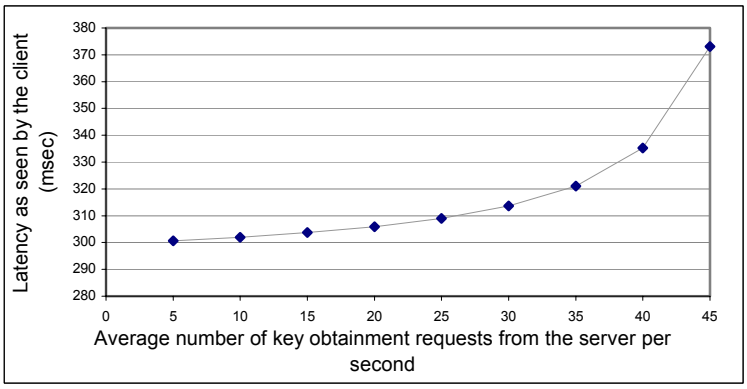

Fig. 5. The change of message verification latency as seen by the client while the average load on the server increases

5 Discussions and Conclusions

In this paper, we proposed an e-mail system, named PractiSES, for key distribution and management in secure e-mail processing. In this system, trusted centralized servers store the public keys and distribute them in an authentic way. We have implemented both server and client interfaces of PractiSES using Java programming language. Besides key management and distribution features, client part includes message encryption/decryption and signature/verification functions as well. PractiSES Server distributes public keys on demand by sending out the requested public keys

with its signature on it. It is a practical key distribution mechanism that does not require revocation control at the client side.

PractiSES takes advantage over S/MIME applications by escaping use of certificates for masses and over PGP by less complicated trust mechanism based on a trusted third party.

Every key operation, even initialization, is performed in an online manner in PractiSES. Moreover, decryption, signature verification and key obtainment services are performed transparent to the users. In other words, the client module senses the security requirements of each message and responds to those requirements automatically. This response is an intelligent one too; e.g. if the recipient cannot process a digitally signed e-mail, the sender's client module detects this by talking to the server and sends a normal message instead of a signed one.

Disclosure of e-mail addresses and other personal information in an uncontrolled fashion is not a problem of PractiSES.

Fortunately in PractiSES, it is not necessary to employ an extra mechanism for revocation control of end user public keys. Revocation of an end user public key in PractiSES is as easy as a database record update.

Our simulations show that a state-of-the-art PC can be used as the domain server for a medium-size organization with up to a traffic of one million e-mail messages per day. For larger organizations with higher e-mail traffic, it is always possible to use a more powerful server or several replicated servers that share the load. Thus the scalability problem of PractiSES due to load on server is not more than any server based application and thus tractable with a proper investment.

References

1. Ramsdell, B. (editor), S/MIME Version 3 Message Specification, RFC 2633, June 1999.
2. Network Associates, "PGP Freeware for Windows 95, Windows 98,Windows NT, Windows 2000 & Windows Millennium User's Guide Version 7.0", available from http://www.pgpi.org/doc/guide/7.0/en/win/, 2001.
3. Housley, R., W. Ford, W.Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL profile", RFC 2459, 1999.
4. Stallings, W., "Cryptography and Network Security, 3/E", Chapter 11, Prentice Hall, 2003.
5. J. Jonsson, B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
6. National Institute of Standards and Technology (NIST), "FIPS Publication 46-2: Data Encryption Standard", 1993.
7. Rivest, R., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Comm. of the ACM, vol. 21, no. 2, pp. 120-126, Feb. 1978.
8. National Institute of Standards and Technology (NIST), "FIPS Publication 180-1: Secure Hash Standard", 1995.
9. Krawczyk, H., M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC2104, 1997.
10. Levi A., "How Secure is Secure Web Browsing", Comm. of the ACM, vol. 46, no. 7, pp. 152, July 2003.