Network Working Group
draft-brown-pgp-pfs-00.txt
Updates: RFC 2440
Category: INTERNET-DRAFT
Expires: 16 January 2001

I. Brown
University College London
A. Back
Zero-Knowledge Systems
B. Laurie
A.L. Digital Ltd
July 2000

Forward Secrecy Extensions for OpenPGP

Status of This Memo

Copyright

Abstract

The confidentiality of encrypted data depends on the secrecy of the key needed to decrypt it. If one key is able to decrypt large quantities of data, its compromise will be disastrous. This memo describes three methods for limiting this vulnerability for OpenPGP messages: reducing the lifetime of confidentiality keys; one-time pads; and the additional use of lower-layer security services.

Table of Contents

1. Introduction

OpenPGP systems [1] allow two strangers to communicate privately. Each user has a public key that is widely disseminated, and a private key that they keep secret. A message encrypted with a public key can only be decrypted with the related private key. The confidentiality of all messages encrypted with a public key rests on the secrecy of the associated private key.

Online systems such as Secure IP [2] can negotiate new keys for every communication using an algorithm like Diffie-Hellman [3]. If a key is compromised, only the specific session it protected will be revealed to an attacker. This desirable property is called perfect forward secrecy. The security of previous or future encrypted sessions is not affected. Keys are securely deleted after use. Without these keys, there is no way captured ciphertext can be decrypted.

It is more difficult to make store and forward systems like e-mail forward secret, as they rarely make direct connections between a message sender and its recipient. In a typical e-mail encryption system, users create a long-term key pair and publish the public key in a directory, on their Web page, or via other methods. While the use of long-term keys reduces the administrative burden of key distribution, the practice introduces vulnerabilities. If a public key is used for several years, as is common with OpenPGP systems, compromise of the private key will allow an attacker to decrypt any message captured during that time.

This memo describes several methods of reducing the vulnerabilities introduced by use of long-term keys. They are a series of options that MAY be implemented by OpenPGP clients for increased security.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Short-lifetime encryption keys

Using a series of encryption keys, each with a short lifetime, reduces the information revealed by the compromise of any one private key because each key protects less data. If expired keys are securely deleted, attackers will never be able to retrieve them to decrypt captured ciphertext. Therefore when a public encryption key expires, an OpenPGP client MUST securely wipe the corresponding private key [4,5].

Deletion should take place once all messages that could have been sent before expiry have been received and decrypted. For example, as a user logs on, their mail client SHOULD retrieve and decrypt all messages from their mail server before deleting any newly-expired private keys. A "panic mode" MAY bypass this step.

PGP clients are able to group "subkeys" together under a long-term signature key to signify their common ownership by one principal. To simplify key management, short lifetime keys SHOULD be created as subkeys of their owner's long-term signature key.

Clients MAY warn senders of messages encrypted with an expired key that they should not use that public key again.

Clients MUST warn all senders of messages encrypted with a revoked key that they should not use that public key again. Any relevant key revocation certificates MUST be included in the warning.

Some OpenPGP systems currently store original message ciphertext and decrypt only for display. While this protects messages on disk, it means that keys must be stored until all messages they protect are deleted. We must assume that an attacker has copies of message ciphertext sent over an insecure network such as the Internet. These messages remain vulnerable until the corresponding private key is deleted.

Messages therefore MAY be stored temporarily encrypted with a short-lifetime key, but NOT any longer than the key's lifetime; they are unreadable once it has been deleted. Clients MUST allow messages to be stored encrypted under a long-term storage key. A mail client MAY implement its own secure storage facilities, or use those provided by other software.

2.1 Key generation and distribution

There is a trade-off for the user: the cost of generating and distributing a new encryption key against the security advantage obtained by earlier key expiry.

Key generation is typically a time-consuming operation. The client SHOULD minimise the time required by the user to complete this operation. This can be achieved, for example, by background key generation, or by using trade-offs that speed up key generation with minimal reduction in security. With Elgamal [6], for example,

the expensive key component to generate is the public prime modulus. A group of keys can share a common public modulus with no negative security implications other than that the key then presents a fatter target for pre-computation attacks. Multiple forward secret Elgamal keys MAY therefore use the same prime modulus with minimal security reduction.

Key distribution can be eased by submitting new keys to key servers, where they will be available for other users to retrieve. Submission and retrieval of generally-available public keys SHOULD be performed automatically by software. Expired public encryption keys MAY be deleted by users and keyservers to save space.

If an OpenPGP client has more than one valid encryption key available for a given message recipient, the key nearest its expiration date MUST be used. This limits the time during which the corresponding private key will be available to an attacker. The time required to deliver a message should be taken into account when checking an expiry date.

Signature keys that are long-lived and certified by other users allow a web of trust to build up. Encryption keys SHOULD be certified by a user's long-term signature key to allow their verification by other users.

2.2 Key surrender

Before an OpenPGP client exports a private key as plaintext, the associated public key MUST be revoked and redistributed. A "reason for revocation" signature subpacket MUST be included in the key revocation specifying "Key material has been compromised" (value 0x02).

3. One-time keys

Taking short-term keys to their logical conclusion, a different key could be used to protect every message. Schneier and Hall [7] suggested a user could make several public keys available in a directory. After a key was retrieved by another user, it would be deleted. This requires message senders to have online access to a directory. Not all e-mail users have this facility. It also allows an attacker to mount a denial of service attack by exhaustively requesting new one-time keys from the directory.

An off-line scheme is more compatible with the store and forward nature of e-mail, and resistant to DoS. Every time a user sends a message encrypted with a public key whose signature includes a one-time key support subpacket, they SHOULD include a new one-time

public subkey for the recipient to encrypt any reply with. It MUST be sent in the format [primary public signing key | one-time public subkey | signature by primary key]. If PGP/MIME [8] support is available, new key(s) MUST be sent in a separate application/pgp-keys MIME bodypart.

One-time subkeys MUST NOT be exported by their recipient to a third party, particularly a key server.

Users still MUST possess a relatively long-lived encryption key. If Alice were writing to Bob for the first time, she would encrypt her message with his long term key. She would also include a newly created one-time public subkey. Bob would use this new key the next time he wrote to Alice, then wipe it. Alice would decrypt the message with the associated private subkey, then delete it.

A "one-time key support" signature subpacket on a public key indicates support for one-time keys. These subpackets are formatted as follows:

Subpacket length: 1
Subpacket type: 30

One-time key support subpackets MUST be included in the hashed area of a signature.

A "one-time key" signature subpacket marker MUST be present in the signature of a one-time subkey. These subpackets are formatted as follows:

Subpacket length: 1
Subpacket type: 158

One-time key subpackets MUST be included in the hashed area of a signature. They are marked as critical so that the entire signature will be ignored by non-compliant OpenPGP clients, preventing more than one message being encrypted using a one-time subkey.

When encrypting messages to a key with a signature containing a one-time key support subpacket, at least one new public encryption subkey MUST be included in the message. This key MUST be signed by the sender's long-term signature key and include a one-time key signature subpacket. It MUST have a short lifetime of less than 30 days, beyond which time the recipient is unlikely to reply to the message. This minimises the key storage requirements of sender and recipient. As a user's collection of private keys grows, she may wish to reduce the lifetime of new one-time subkeys.

A client MUST include further new public encryption subkeys if it believes a message will receive multiple replies, each of which SHOULD be encrypted with a different subkey if available.

Clients MUST delete a one-time subkey after successfully encrypting data using it. They SHOULD use a one-time subkey, if available, in

preference to short-lifetime key.

4. One-time pads

The one-time pad is the only provably secure cipher [9]. It uses a secret key as long as the plaintext; the key is XOR'd with the plaintext to give the ciphertext, and with the ciphertext to give the plaintext. Each secret key MUST only be used once. Compromise of key data allows decryption only of the matching ciphertext. A key can always be generated that will create any given plaintext of the same length from a piece of ciphertext.

While the OTP may be considered overkill for most OpenPGP applications, particularly given the far greater insecurities present in common applications and operating systems, it can be useful for ultra-secure communication between two parties who have already securely physically exchanged key material out of band. OTP keys MUST NOT be transferred by a less secure method, for example using any other cipher. A OTP SHOULD only be used in a trusted computing environment: to do otherwise gives a false assurance of security.

4.1 One-time pad storage

One-time pad data MUST be securely transferred out-of-band. An application SHOULD store pad data in a Literal packet (tag 11). The body of the packet consists of:

- One octet 'o' (0x6f) specifying binary one-time pad data.
- One octet 0x00 specifying no file name.
- A four octet date specifying the creation time of the packet.
- The one-time pad.

4.2 One-time pad reference

A one-time pad reference is used to tell a message recipient which pad data to use to decrypt the following encrypted packet. It is stored in a Symmetric-Key Encrypted Session-Key Packet (Tag 3).

The body of this packet consists of:

- A one-octet version number. The only currently defined version is 4.
- A one-octet number describing the symmetric algorithm used: 14.

- 0x0000 to specify that the reference is not encrypted.

- A four-octet date when the referenced one-time pad was created.

- A four-octet offset specifying the first octet in the referenced pad that should be used as key.

4.3 One-time pad encryption

Data encrypted with a one-time pad is stored in a Symmetrically Encrypted Data Packet (Tag 9) using a cipher value of 14. Its body is simply a valid OpenPGP message (typically consisting of literal or compressed data packets) exclusive-OR'd with the one-time pad data referred to by the preceding one-time pad reference.

Once a message recipient has decrypted a one-time pad message, it MUST securely delete the one-time pad data used.

Clients SHOULD use a one-time pad, if available, in preference to one-time or short-lifetime keys.

5. Secure and decentralised e-mail transport

The vast majority of current mail clients deliver messages first to a local mail server, which forwards them to their recipient's mail server, where they remain until collected. This procedure minimises security because it fails to take advantage of mail transport protocols such as SMTP [10] over secure transport or network layer security links such as TLS [11] or IPSEC [2]. This is particularly important given that these protocols allow transient keys to be generated and then discarded after each session, providing perfect forward secrecy.

End-to-end security would be better provided if clients delivered messages directly to the recipient's mail server. This allows a secure link to be set up between the two, providing a second layer of forward secrecy. Ideally, as greater numbers of users gain permanent Internet connections through cable modems or Digital Subscriber Lines, they can run mail servers on their own machines. DNS Mail eXchange records [12] can be used to specify a backup mail server such as at an ISP for times when the recipient's machine is unavailable.

OpenPGP mail clients SHOULD deliver messages directly to the recipient's mail server, and MUST use any available lower layer security services to protect the links used to deliver messages.

Where OpenPGP keys are used in such services, they SHOULD NOT be used to encrypt keying material that can later be decrypted if they are compromised. Ideally, they SHOULD be used only to authenticate a forward-secret key negotiation protocol such as Diffie-Hellman [3]. At the least, new short-lifetime key pairs SHOULD be generated for key encryption use.

Direct delivery of mail can reveal the sender and recipient of

messages to traffic analysts. Clients MAY use anonymous remailers [13] or IP [14] services to mask this information.

6. Security Considerations

As mentioned in section 4, users of these extensions must consider the complete security environment in which they are operating. Highly-secure communications are of limited use between two insecure systems vulnerable to hackers, virii, and other methods of message and key compromise at source. Bellovin [15] describes a minimum set of precautions that should be taken.

7. Acknowledgements

Thanks to Nick Bohm, Richard Clayton, Hal Finney and Edwin Woudt for suggestions that have been incorporated into this draft.

8. Authors' Addresses

Ian Brown
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom

Phone: +44 20 7679 3716
Fax: +44 20 7387 1397
E-mail: I.Brown@cs.ucl.ac.uk

Adam Back
Zero-Knowledge Systems Inc.
888 de Maisonneuve East
6th Floor
Montreal
Quebec H2L 4S8
Canada

E-mail: adam@cypherspace.org

Ben Laurie
A.L. Digital Ltd.
Voysey House
Barley Mow Passage
London W4 4GB
United Kingdom

Phone: +44 (20) 8735 0686
E-mail: ben@algroup.co.uk

9. References

[1] Callas, J., Donnerhacke, L., Finney, H. and Thayer, R., "OpenPGP Message Format", RFC 2440, November 1998.

[2] Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825, August 1995.

[3] Diffie, W. and Hellman, M., "New directions in cryptography", IEEE Transactions on Information Theory 22(6), November 1976, 644-654.

[4] US Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria", DoD 5200.28-STD, December 1985.

[5] Crescenzo, G. de, Ferguson, N., Impagliazzo, R. and Jakobsson, M., "How To Forget a Secret", Proc. Symposium on Theoretical Aspects in Computer Science, Trier, Germany, March 1999.

[6] Elgamal, T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory 31(4), July 1985, 469-472.

[7] Schneier, B. and Hall, C., "An Improved E-mail Security Protocol", Proc. 13th Annual Computer Security Applications Conference, New York: ACM Press, 1997, pp. 232-238.

[8] Elkins, M., Del Torto, D., Levien, R. and Roessler, T., "MIME Security with OpenPGP", IETF work in progress, April 2000.

[9] Schneier, B., "Applied Cryptography", New York: Wiley, 1996, p.15.

[10] Postel, J., "Simple Mail Transfer Protocol", RFC 821, August 1982.

[11] Dierks, T. and Allen, C., "The TLS Protocol", RFC 2246, November 1997.

[12] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[13] Chaum, D., "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", Communications of the ACM 24(2) 84-88, February 1981.

[14] Reed, M. G., Syverson, P. F. and Goldschlag, D. M., "Anonymous Connections and Onion Routing", IEEE Journal on Selected Areas in Communication Special Issue: Copyright and Privacy Protection, May 1998.

[15] Bellovin, S., "Can Someone Read My E-Mail?", http://www.research.att.com/~smb/securemail.html, 1998.

10. Full Copyright Statement