algo^{logic}

Algologic Research and Solutions
Technical Report

Surreptitious forwarding – a subtle form of fraud

May 2013

algo^{logic}

Algologic Technical Report # 01/2013

Surreptitious forwarding – a subtle form of fraud

S. Parthasarathy
drpartha@gmail.com

Our Ref.: surreptitious.tex
Vers. code: 20130505

Abstract

This tutorial report exposes the risks involved in reckless combination of otherwise potent cryptographic tools. It is inspired from [1] [2]. It gives simplifed examples of such risks, and concludes with a warning concerning the usage of enigmail/Thunderbird.

1 Description of the problem

Let us take a scenario involving two parthers who want to exchange some information (message). In real life, this act is fraught with many risks and hasards. Cryptography offers two tools for reducing the risks, and ensuring security of messages:

Encryption (E): Only the recipient can decrypt the message (and modify it, if needed).

Signing (S): The recipient knows who wrote the message. Modifications if any, made to the message, after signing, are easily detected.

To ensure a greater level of protection, it is often a practice to both sign and encrypt (S & E) sensitive messages. Naïve S &E can lead to a new form of risk to the communication between sender and receiver. The combination of two cryptographic techniques – encryption, and digital signature, can lead

to a subtle form of fraud. This kind of fraud does not involve exposure of confidential contents, or tampering of confidential data, yet, it can lead to embarassing and damaging consequences. We call this "surreptitious forwarding" [1]. This form of fraud is a variant of what is commonly called as "replay attack" [6].

We take a scenario involving two parthers who want to exchange some sensitive information (message): Sita and Rama [3]. Sita wants to send a message to Rama. Hoping to avoid any mischief, Sita signs & encrypts (S & E) the message for Rama's eyes, but Rama re-encrypts Sita's signed message for Tommy to see. In the end, Tommy believes Sita wrote to him directly, and can't detect Rama's subterfuge.

In the rest of this article, we will assume that Sita and Rama use "asymmetric key" cryptography [4]. In this approach, two keys are used to encrypt (and subsequently decrypt) a given message. The sender use the "public key" of the receiver, to encrypt the message. The receiver uses his/her own "private key", to decrypt the message. The sender can digitally "sign" the message, for which, he/she uses his/her own secret key. The receiver can "verify" the signed message, using the public key of the sender. We use the following notation (inspired from [1]), in this article :

msg – raw message

$\{\text{msg}\}_a$ – raw message, signed using the private key of the sender a

$\{\text{msg}\}^B$ – raw message, encrypted using the public key of the sender b

}encmsg{ – decryption of the encrypted message encmsg. Yields the raw message msg

We examine three examples of surreptitious forwarding :

- Sita wants to say "I love you" to Rama. {I love you} is the initial message, intended for Rama. Sita digitally signs the message $\{\text{I love you}\}_s$, and then encrypts it for Rama to see $\{\{\{\text{I love you}\}_s\}^R\}$.

  Rama wants to play a prank on Sita. He decrypts the message he got from Sita and re-encrypts it to Tommy $\{\{\{\text{I love you}\}_s\}^T\}$

When Tommy sees and decrypts $\{\{\{\text{I love you}\}_s\}^T\}$, he gets the impression that Sita is in love with him (Tommy) ! Tommy will feel disappointed when he comes to know the truth.

- Sita wants to send a secret formula "The secret formula is XCV76B" to Rama. Sita digitally signs the message $\{\text{The secret formula is XCV76B}\}_s$, and then encrypts it $\{\{\{\text{The secret formula is XCV76B}\}_s\}^R\}$.

  Rama wants to play a prank on Sita. He decrypts the message he got from Sita and re-encrypts it to Tommy $\{\{\{\text{The secret formula is XCV76B}\}_s\}^T\}$.

  When Tommy sees and decrypts $\{\{\{\text{The secret is XCV76B}\}_s\}^T\}$, he gets the impression that Sita is betraying her employer by revealing company secrets to him (Tommy) ! Sita may even lose her job because of this.

- Sita wants to send some moeny to Rama. Sita digitally signs an electronic cheque $\{\text{Pay Rs. 10 000 to the bearer}\}_s$, and then encrypts it $\{\{\{\text{Pay Rs. 10 000 to the bearer}\}_s\}^R\}$.

  Rama wants to play a prank on Sita. He decrypts the message he got from Sita and re-encrypts it to Tommy $\{\{\{\text{Pay Rs. 10 000 to the bearer}\}_s\}^T\}$

  When Tommy sees and decrypts $\{\{\{\text{Pay Rs. 10 000 to the bearer}\}_s\}^T\}$, he gets the impression that Sita wants to pay him (Tommy), and encashes the cheque ! Sita will later have a lot of difficulty explaining why she wrote multiple cheques without adequate justification.

Interestingly, naïve Encrypt-then-Sign (E & S) isn't any better than Sign & Encrypt (S & E) either. In this case, it is easy for any eavesdropper to replace the sender's signature with his own, so as to claim authorship for the encrypted plaintext.

2 Remedial measures

The two approaches : encryption, signing, have their own limitations. Encryption alone, does not register as to who did the encryption. Signing alone,

does not specify as to who the authorised destinator is. There is no coupling between the two approaches when they are used in cascade.

Naïve Sign & Encrypt has come to characterize file-handling and e-mail security applications. PKCS#7, CMS, S/MIME, and PGP, all suffer from this class of vulnerabilities [1][2]. The first version of the PGP message-format , and X.509v1 have also been reported to suffer from similar weaknesses. [1][2] describes in detail, the vulnerabilities involved. According to [1][2], these standards all erred by treating public-key encryption and digital signatures as if they were fully independent operations. This independence assumption, although convenient for writing standards and for writing software, is cryptographically incorrect. When independent operations are applied, one on top of another, then the outermost crypto layer can undetectably be replaced, and security is weakened.

The real weakness is not in the individual approaches, encryption or signing, but rather in an inappropriate usage in tandem of the two approaches. Actually, this is manifested only in the interpretation of the end result after the combined usage. Cascaded usage implies that the two independent applications are decoupled or isolated from each other. A simple remedy works on the principle that the decoupling should be broken. In order to work properly together, the signature layer and the encryption layer actually must refer to one another. The recipient needs proof that the signer and the encryptor were the same person, which necessarily entails either signing the recipient's identifier (in Sign & Encrypt), or encrypting the signer's identifier (in Encrypt & Sign). Once such cross-references are in place, an attacker cannot remove and replace the outermost layer, because the inner layer's reference will reveal the alteration. This can be achieved only be enforcing a discipline in the usage of these two approaches.

One aspect of cascaded usage of signing and encrytion remains unresolved fully. It is not easy to pinpoint the legal culpabibilty of the persons concerned, since no malicious tampering of contents, or distorsion was ever done. To complicate matters, the victim of such infractions, is not always the same person.

3 A case study

Enigmail is a security extension to Mozilla Thunderbird and Seamonkey [7]. Enigmail acts as a front end to OpenPGP. It allows a user to easily sign

and encrypt a message (using OpenPGP), from within Thunderbird, before sending the message. This facility is an invitation to disaster, since enigmail does not issue any warning or alert to the sender, when such a combination is chosen by him/her. Neither does it flag the message, so as to alert the person who receives the message with a potential bomb concealed in it. Users of enigmail are therefore advised to take appropriate precautions.

4 Concluding remarks

This tutorial article borrows heavily from [1] and [2].

This is a LaTeX document, created under Linux, using Kile. You can get the LaTeX source of this document from drpartha@gmail.com. Please mention the Reference Code, and Version code, given at the top of this document.

If you found this article useful, please send a note to `drpartha@gmail.com` As always, suggestions and constructive comments are always welcome.

References

[1] Donald T. Davis, *Defective Sign & Encrypt in S/MIME, PKCS# 7, MOSS, PEM, PGP, and XML.*,
Proc. Usenix Tech. Conf. 2001 (Boston, Mass., June 25-30, 2001), pp. 65-78.

[2] Donald T. Davis, *Defective Sign & Encrypt in S/MIME, PKCS# 7, MOSS, PEM, PGP, and XML.*
`http://world.std.com/\~dtd/sign_encrypt/sign_encrypt7.html`

[3] Parthasarathy S, *Alice and Bob can go on a holiday*,
`http://profpartha.webs.com/publications/alicebob.pdf`

[4] Bruce Schneier

[5] Creative Commons, *By Attribution - NonCommercial - ShareAlike 3.0 Unported License*,

http://$creativecommons.org/licenses/by-nc-sa/3.0/deed.en_US$

[6] William Stallings, *Cryptography and network security*, Pearson Education, Delhi, India

[7] enigmail, *A simple interface for OpenPGP email security*, http://www.enigmail.net/home/index.php

* * * *