Mutual Authentication Protocol For Safer Communication

S P Shantharajah K Duraiswamy

Abstract

With a network communication requires cryptography authentication for secure transmission. This paper takes two challenge-response protocols by which entities may authenticate their identities to one another in a mutual communication way. Here the two entities, each holding a share of a decryption exponent, collaborate to compute a signature under the corresponding public key. These may be used during session initiation, and at any other time that entity authentication is necessary. The authentication of an entity depends on the verification of the claimant's binding with its key pair and the verification of the claimant's digital signature on the random number challenge. The network defined mutually authenticated protocols are for entity authentication based on public key cryptography, which uses digital signatures and random number challenges. Authentication based on public key cryptography has an advantage over many other authentication schemes because no secret information has to be shared by the entities involved in the exchange. This paper analyzes the protocols, which minimizes cost, very simple and suggests security for mutually signing the signature schemes and provides proofs of security for the safer communication. This paper specifies the way of mutual communication and the method of processing that conversation between entities.

Keywords : Authentication, Computer Security, Cryptographic Modules, Cryptography, Digital Signatures, Proofs of Security.

0. Introduction

Authentication based on public key cryptography has an advantage over many other authentication schemes because no secret information can be shared by the entities involved in the exchange. A user (claimant or Initiator or A) attempting to authenticate oneself must use a private key to digitally sign a random number challenge issued by the verifying entity. This random number is a time variant parameter, which is unique to the authentication exchange. If the verifier can successfully verify the signed response using the claimant's public key, then the claimant has been successfully authenticated.

Here, consider a signature scheme of the "hash-then-decrypt" variety, meaning the public key is N, e, the secret key is d, and the signature of message M is $H(M)^d$ mod N, where H is a public hash function, N is an RSA modulus, e is an encryption exponent, and d is the corresponding decryption exponent. However, instead of there being a single signer, the public key is associated to a pair of entities (even that termed to be a initiator and a responder or a client and a server). The decryption exponent *d* is not held by any individual party, but rather is split into shares d_c and d_s, and these are held by the two entities i.e. the initiator and the responder respectively. A collaborative computation, or signing protocol, is used to produce a signature for the receiver which leads to authentication.

1. Computation of Collaborative Signature

RSA, due to its algebraic properties, lends itself naturally to collaborative signature computation. The decryption exponent is split multiplicatively, meaning

$d_c d_s$ *aH* d (mod $ö(N)$)

Collaborative signature computation is then based on the equation

$H(M)^d$ *aH* $H(M)^{dcds}$ mod N

This paper considers a natural and simple signature schemes based on direct exploitation of the second Equation (above). We ***divide them into two classes***. In the *common-message* class of schemes, the message M to be signed is a common input to an initiator and responder i.e. client and server. Within this class we consider two protocols:

MCS: Client sends $x_c = H(M)^{dc}$ mod N to the server;
Server computes signature $x = X_c^{ds}$ mod N,
Verifies it, and returns it to client.

MSC: Server sends $x_s = H(M)^{ds}$ mod N to client;
Client computes signature x = x_s^{dc} mod N. 1

(In the terms MCS, MSC states à M-,Message from, C-Client and S-server)

The leading "M" in the common-message protocols is the "Message" that both entities know. In the client-message class of schemes, the message M to be signed is input to the client but not to the server. Within this class we again consider two protocols for computing a "partial" signature:

HCS : Client sends y, x_c to the server,
Where $y = H(M)$ and $x_c = y^{dc}$ mod N ;
Server computes signature $x = x_c^{\ ds}$ mod N,
Verifies that x^e *a"* y (mod N) and returns x to client.

HSC : Client sends y to the server, where $y = H(M)$;
Server sends $x_s = y^{ds}$ mod N to client;
Client computes signature $x = x^{dc}_{\ s}$ mod N.

(Similarly, in the terms HCS, HSC states à H-, Hash from, C-Client and S-server)

Here the leading "H" in the names of the client-message protocols stands for the "Hash" that the client flows to the server. The other letters reflect the order in which the Client and Server use their shares of the decryption exponent in the protocol.

In this paper we analyze the security of the above four protocols with regard to meeting well-defined modern cryptographic goals in provable ways. We find that the security goals, and the assumptions on the underlying primitives required to prove security, vary from protocol to protocol in a perhaps surprising way.

2. Security Analyses

The first security goal that comes in consideration is to prevent forgery by a third party. We suggest however that this goal is too weak, and instead ask that forgery be hard even for an adversarial client who is in possession of the correct share d_c and is allowed to engage in interactions with the d_s-holding server. Security against third parties is implied by security against client adversaries, so consideration of the latter only strengthens the security results. This is appropriate because we view the server is a "co-signer" of the client. A verifier who accepts a client signature does so under the belief that the server "*endorses*" it. (A client who succeeds in creating a signature that the server has not endorsed should be viewed as having been successful in forgery.)

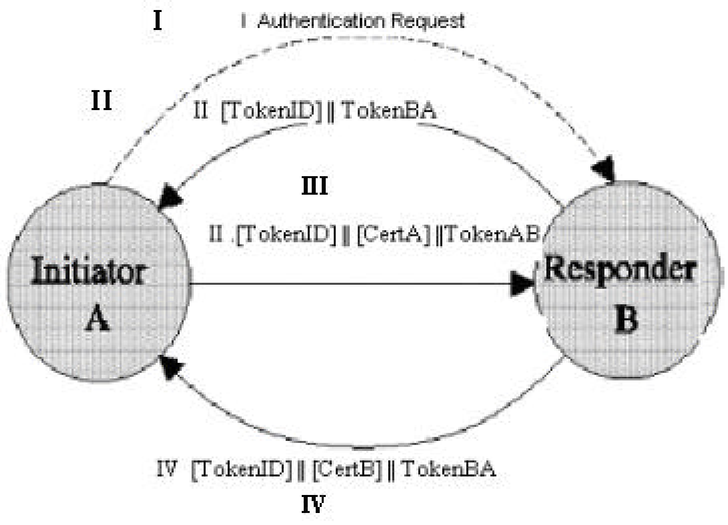

Diagram –1. Authentication process

2.1. Mutual authentication protocol

The mutual entity authentication protocol said to be "Three pass authentication" is given in the above diagram(Diagram –1). Certain authentication token fields and protocol steps are specified in greater detail in this paper. Either entity may choose to terminate the authentication exchange at any time. The mutual authentication protocol refers to two entities (say "initiator" - A and "responder" - B). Here each entity acts as*both* a claimant and a verifier in the protocol (i.e. an initiator and a responder). It is important to note that the success of an entity's authentication, according to this standard, is not dependent on the information contained in the text fields.

The authentication of an entity depends on two things:

1. The verification of the claimant's binding with its key pair,
2. The verification of the claimant's digital signature on the random number challenge.

Authentication occurs as follows:

I. The initiator, A, selects the responder, B, with which it will mutually authenticate, and makes an authentication request to B

II. The responder, B, determines if it will continue, initiate, or terminate the authentication exchange. If it attempts to authenticate the initiator, the responder then

i. Generates a random number challenge, which is the value for the RB field in Token BA.

ii. Generates and/or selects other data which is to be included in the TokenBA.

The responder creates a challenge token of the following form:

Enc [TokenBA = RB || Text]

where 'Enc' **refers** Encryption.

RB – Random Number of B

Entity B sends a message consisting of TokenBA and an optional TokenID to the initiator.

The message from the responder to the initiator is of the form:

[TokenID] || TokenBA

III. Upon receiving the message including TokenBA , the initiator, A, Uses TokenID to determine which token is being received.

Retrieves information from the Text1 field, using it in a manner, which is outside the scope of this standard.

iii. Generates a random number challenge which is the value for the R field in TokenAB

iv. Selects an identifier for the responder, and includes that in the B field of TokenAB.

The initiator creates an authentication token, TokenAB, by concatenating data and generating a digital signature:

TokenAB = Enc [RA || [RB] || [B] || [Text] || S (RA || RB || [B] || [Text])]

The signed data are present only when their corresponding values are present in the unsigned part of TokenAB, although RB does not have to be in the unsigned data of TokenAB.

In addition to containing TokenAB, the message may include a token identifier, TokenID, and the initiator's certificate CertA. i.e. The message from the initiator to the responder is of the form:

[TokenID] || [CertA] || TokenAB

Upon receiving the TokenAB transmission, the responder, B,

i. Uses TokenID to determine which token is being received.

ii. Verifies that the value of RB which is present in the unsigned part of TokenAB.

iii. Verifies the initiator's certificate and verifies the initiator's signature in TokenAB.

Successful completion of this process is that the initiator, A, has authenticated itself to the responder,

B. If any of the verifications fail, then the authentication exchange is terminated.

The responder, B,

(i) Selects an identifier for the initiator, and includes that in the A field of TokenBA

(ii) Generates and selects other data which is to be included in the Text fields. In TokenBA, Text is a subset of the Text field.

The responder creates an authentication token, TokenBA, by concatenating data and generating a digital signature:

TokenBA = Enc[[RA] || [RB] || [A] || [Text5] || S (RA || RB || [A] || [Text4])]

The responder to the initiator is of the form:

[TokenID] || [CertB] || TokenBA

Upon receiving the message including TokenBA, the initiator, A,

i. Uses TokenID to determine which token is being received.

ii. Verifies that the value of R

iii. Verifies that the identifier for the responder has been obtained in CertB, TokenBA.

iv. Verifies the responder’s certificate

v. Verifies the responder’s signature on TokenBA.

Successful completion is that the responder, B, has authenticated itself to the initiator, A, and thus the entities have successfully mutually authenticated.

4. Conclusion

From the points discussed above, we can conclude with security issues which state that, by assuming the base signature as secure one, we make the mutual authentication more secure one. A responder (B) can identify a initiator (A) in a safer way so that nobody can impersonate any one. The additional security is that no hackers can intrude into their communication area. During communication the verification is done every time by both the entities and leads to secure transmission against hackers attack.

Here both the entities are identified by each other by performing cryptographic computations, validations and verification. Finally, the mutually authentication protocols provides authentication between the entities.

5. References

1. M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko ,(2001), The one-more-RSA-Inversion problems and the security of Chaum’s blind signature scheme. Lecture Notes in Computer Science Vol.2330, P. Syverson ed., Springer-Verlag.
2. A. Yao, (1986), How to Generate and Exchange Secrets, Proceedings of the 27th Symposium on Foundations of Computer Science, IEEE.
3. S. Goldwasser, S. Micali and R. Rivest, (April 1988) A digital signature scheme secure against adaptive chosen-message attacks, SIAM Journal of Computing, 17(2); 281-308.
4. D. Chaum, (1983), Blind Signatures for untraccable payments. In Advances in Cryptology – CRYPTO’82, Plenum Press.
5. ITU-T Rec. X.509 | ISO/IEC 9594-8, (1993), Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, Editor’s DRAFT.

About Authors

S. P. Shantharajahis a Lecturer in Department of Computer Science in K.S.R. College of Technology, Tiruchengode, Namakkal Dt., Tamilnadu.
E-mail : spshantha_raj@yahoo.com

Dr. K. Duraiswamyis a Principal in K.S.R. College of Technology, Tiruchengode, Namakkal Dt., Tamilnadu.