Deniable Authentication with RSA and Multicasting

Daniel R. L. Brown*

February 24, 2005

Abstract

A deniable authentication scheme using RSA is described and proven secure in the random oracle model. A countermeasure to a well-known attack on efficient deniable authentication to multiple recipients is described and proven secure.

1 Introduction

Email and text messaging are single-pass media. Their authentication has therefore traditionally used digital signatures. But signatures leave evidence that third parties can verify. Sometimes this is undesirable and it is preferable to have deniable authentication instead. Indeed, while common sense dictates to authenticate all messages, it also dictates to be very careful what you sign.

In deniable authentication, a sender Alice and a recipient Bob each have their own public keys. Alice sends an authenticated message to Bob using her private key and Bob's public key. Bob verifies the authenticated message with his private key and Alice's public key. Alice does this without digitally signing anything. Bob has no proof to others that Alice used her private key to do anything.

Related concepts to deniable authentication are *plausible deniability* and prevention of *surreptitious forwarding*. The IETF S/MIME protocol, which can be used to secure email, includes an *AuthenticatedData* type that does not include a signature, and is instead secured with Diffie-Hellman key agreement [Hou99] or elliptic curve Menezes-Qu-Vanstone (ECMQV) [BWBL02]. The term *deniable authentication* is also used in [dRG05] for essentially the same notion as here. Less similar concepts are *undeniable signatures* and *designated confirmer signatures*.

Deniable authentication is straightforward with Diffie-Hellman (DH) key agreement and its derivatives such as ECMQV, at least for the case of single recipients. Most (single-pass) authentication methods based on Rivest-Shamir-Adleman (RSA) have, however, used a conventional signature, and as such were not fully deniable. A deniable authentication scheme based on RSA is described and proven secure under the standard RSA assumption and the random oracle model. It is also proved secure under a strong assumption about RSA and a milder assumption on hash functions.

In efficient multi-recipient deniable authentication, it is necessary not to permit a malicious recipient to modify the message for other recipients. This paper provides a mechanism to prevent this kind of attack.

*Certicom Research

1.1 Applications

Deniable authentication can be used as part of a countermeasure against spam. If recipients insist that every message they receive is authenticated, then they can resist sender address spoofing. Once sender identities are reliable, sender address filtering can be more effective.

Against spam, deniable authentication has an advantage over signatures in that senders of mass recipient messages must authenticate separately for each recipient. Spammers thus incur a per-recipient disincentive. This can be combined with other techniques of adding a sender-side cost.

Another application of deniable authentication is the protection of sensitive organization data. Suppose such data is leaked outside the organization. If a signature has been applied to the data, an outsider could ascertain that the data is authentic. If only deniable authentication was applied, however, then the outside could not be sure the leak was not fabricated.

1.2 Standards

Existing standards for discrete logarithm based key agreement, such as X9.42 and X9.63, are already useful for deniable authentication. Indeed, they have already been used in other standards, such as S/MIME, for deniable authentication.

On the other hand, standards for RSA generally do not provide (single-pass) deniable authentication. For example, the current draft of ANS X9.44 gives two key agreement schemes using RSA and one key transport scheme. Both agreement schemes are derived from the protocol TLS. The first key agreement scheme and the key transport scheme do not have bilateral authentication because the sender does not have a public key. The second key agreement scheme involves digital signatures, so full deniability is not possible. Deniable authentication would be a useful addition to standards like ANS X9.44.

Secure efficient deniable authentication with multiple recipients would be a valuable enhancement of standards that specify deniable authentication. The S/MIME standard is good example of this, particularly [BWBL02, Hou99].

1.3 Previous Work

Several key establishment mechanisms using digital signatures and public key encryption are described in [MvOV95, §12.5.2]. These schemes do not have full deniability when used with digital signatures such as PKCS #1 v. 1.5 and PSS. Even when used with raw RSA digital signatures, they do not have full deniability because identifiers are included in the signatures. (Incidentally, raw RSA digital signatures are weak in the sense that existential forgery is easy.)

Triple-wrapping message, either by sign-encrypt-sign or encrypt-sign-encrypt, is not fully deniable authentication, because Alice signs something that Bob can show to others. If the signed data is a ciphertext, Bob can generally reveal the plaintext and provide some additional data such that third parties can verify the correspondence of plaintext to ciphertext. In fact, with most encryption schemes, Bob can do this without even revealing his private key. If the ciphertext is only the encryption a symmetric authentication key, not a message itself, then authentication is weakly deniable, in the sense that Bob cannot prove Alice used the symmetric authentication key on particular messages, but he can prove that Alice authenticated the key.

Some countermeasures to surreptitious forwarding, such as those surveyed by Davis [Dav01], primarily aim to prevent forwarding only in the sense that the user's regular cryptographic application cannot perform it. More active adversaries will not use conventional software, and instead use software that can extract the signatures.

2 Scheme Ingredients

This section defines the ingredients to the scheme.

2.1 Some Notation for General Trapdoor Permutations

The RSA primitive is a *trapdoor permutation*. The deniable authentication scheme described in this paper works for any trapdoor permutation. So, for the sake of generality, we use a generic notation for trapdoor permutations. We first describe how the notation works in general, and then describe how it applies to RSA.

A *trapdoor permutation pair* is (N, n), where N is an easily computable public function and $n = N^{-1}$ is a private inverse function. The function is n is easily computable only by the key pair owner. So, computing n from the description of N is infeasible. Such a function N is called a *trapdoor permutation* and n is called its *trapdoor inverse*. We may also call N and n the public key and private key, respectively, although this clashes slightly with the conventional terminology with RSA.

In the case of RSA, the trapdoor key pair (N, n) is as follows. Let e be some public value, typically $e = 3$ or $e = 2^{16}+1$. Let p, q be secret primes with $\gcd(e, (p-1)(q-1)) = 1$. Let N be the function defined by $N(x) = x^e \bmod pq$. In a slight abuse of our own notation, we also write $N = pq$ in line with conventional notation for RSA. The trapdoor inverse is defined as $n = N^{-1}$, so $n(y) = y^d \bmod N$, where $d = e^{-1} \bmod (p-1)(q-1)$. In another abuse of notation, we might write n for d. The pair (N, n) is an RSA key pair with public permutation N and private permutation n.

Although RSA is the most widely used and known trapdoor permutation, others are known. One of the most important is Rabin-Williams (RW), whose security is known to be equivalent to the hardness of factoring. Our system will work with general trapdoor permutations.

Let $[N]$ be the domain of the function N and let $[n]$ be the domain of function n. For RSA, we can use $[N] = [n] = \{x : 0 \leqslant x \leqslant N-1, \gcd(x, N) = 1\}$, as a subset of integers. In practice, it is equivalent to regard $[N]$ for RSA as the set of integers in the interval $[1, N-1]$.

Both Alice and Bob have key pairs. We write (A, a) for the key pair of Alice, and (B, b) for Bob's key pair. Alice will generally be the sender and authenticator of the message, and Bob will generally be recipient and verifier of authenticated message. Each will have an authentic copy of the other's public key. For further simplicity, we may also identify Alice and Bob with their public keys. The meaning of A and B as entity or public key (as trapdoor permutation or RSA integer) will be made clear, either from context or explicitly.

It will be efficiency in the trapdoor-based deniable authentication scheme for two different domains $[A]$ and $[B]$ to have significant overlap. For RSA functions, this is easily achieved if A and B are near in size.

For other intermediate values in the protocols, we generally use uppercase letters for public values that anybody can determine and lowercase for secret values that only Alice or Bob can determine. This convention is similar to that used with some kinds of public key cryptography.

2.2 Key derivation functions

Established keys are derived from the established secret and other shared data using a key derivation function (KDF). Key derivation functions are constructed from hash functions. One construction is the KDF of ANS X9.63. The schemes described in this paper are such that the following data in the parameters strengthens the security.

- Identifiers for Alice and Bob can be included in the key derivation parameters, which helps to thwart unknown key share attacks. Identifiers can be public keys or representations of names, or hashes of the latter. If symmetry is desired, so that the established key cannot be said to be directed from Alice to Bob, or vice versa, Alice and Bob's identifier can be combined with a symmetric function such as integer multiplication.
- Time or nonce values included in the key derivation function help to ensure freshness of the established key, which helps to ensure weak implicit entity authentication and known-key security.
- A MAC tag can be include the key derivation function. This helps to avoid message tampering that might be possible in an efficient multi-recipient deniable authentication scheme.

2.3 Intermediate Bijection

The security of the deniable authentication may be enhanced with some secure bijections. The bijections should take a form such as $S : \beta \to \alpha$ where $\beta \subseteq [b]$ and $\alpha \subseteq [a]$, with these subsets containing almost elements of their supersets. Both directions of the bijection should be easily computable. The bijection is a fixed public algorithm, although it may optionally have a key. The bijection needs to be secure in a sense similar to a secure hash function. So finding u and $S(u)$ with a given a structure (independent of the definition of S) should be roughly as hard as doing so if S were a random bijection. For example, it should not be possible to make both u and $S(u)$ small. In the security analysis this helps to avoid an attack and to prove security.

The secure bijection helps eliminates structure, such as small size, that an attack might exploit. Potentially, the bijection can be built from a block cipher, or possibly a key wrap function. The key can be fixed for all applications of the scheme, or it could be selected dynamically, or made a function of the other values in the scheme.

2.4 Message Authentication Code

A *Message Authentication Code* (MAC) is an algorithm that takes input of a message M and of secret key K, and then outputs a tag $T = \mathrm{MAC}_K(M)$. Without knowledge of K, it should infeasible to compute the correct value T. In other words, a MAC is believed to be unforgeable. Parties that know K can send a MAC tag T with the message M and then be sure that no other parties have modified the message M.

3 RSA-Based Schemes for Deniable Authentication

Deniable authentication is fairly straightforward with DH based key agreement, as follows. Agree on a symmetric key, using a scheme that provides mutual authentication. Then apply

a (symmetric) message authentication code (MAC) to the message. This is the approach taken in S/MIME's AuthenticatedData type. Essentially, the scheme here is the same. The interesting part is how RSA key agreement is done, in a single-pass, without signing anything.

A simplified version of the scheme is given in Figure 1. Alice sends $Z = a(B(x))$ to Bob, where x is a random value that Bob can recover from Z as $x = b(A(Z))$. Alice and Bob can use x to derive a symmetric key k that they can use for any purpose, generally. In this case, they use k to compute MAC tag T on a message M. Alice sends M, T, and Z in a single pass. In selecting x, Alice uses a while loop to get an intermediate value $Y = B(x) \in [a]$, so that she can apply a. The purpose of this to ensure that Z and x do not have an bias that an attacker might be able to exploit.

Alice		Bob
While$(Y \notin [a])$:		
$\quad x \in_R [B]$		
$\quad Y = S(B(x))$;		
$Z = a(Y)$		
$k = \mathrm{KDF}(x, A, B, T)$		
$T = \mathrm{MAC}_k(M)$		
	$\xrightarrow{M,T,Z}$	
		$Y = S^{-1}(A(Z))$
		$x = b(Y)$
		$k = \mathrm{KDF}(x, A, B, T)$
		$T \stackrel{?}{=} \mathrm{MAC}_k(M)$

Figure 1: Simplified version of RSA message authentication

The scheme in Figure 1 can be simplified by choosing S as the trivial identity function, and by not including identifiers in the key derivation function. This simplifications lead to some problems, however as noted below.

3.1 A Forgery Attack Against Small RSA Exponents

When the trapdoor function is RSA with small values of e, such as $e = 3$, and the bijection S is the trivial identity function, the following attack is possible.

Eve selects a value $Z < \sqrt[e]{A}, \sqrt[e]{B}$. From this Z, Bob will computes $x = b(S^{-1}(A(Z))) = b(Z^e) = Z$. Eve knows $x = Z$, and can derive k and compute $T = \mathrm{MAC}_k(M)$. She can thus forge Alice's authentication.

(If Alice chooses x randomly, there is negligible chance that she will generate such Z.)

Although a suitable bijection S can easily prevents the forgery attack for small exponents, alternative countermeasures are possible. Larger exponents could be used or Bob could just automatically reject small values of Z.

3.2 An Unknown Key-Share Attack

The simplified scheme is vulnerable to an unknown key share attack, which is a kind of an identity theft attack. In an unknown key-share attack, Eve replaces Alice's identity with

her own, making Bob think that Eve sent the message. (If Eve cannot stop the message from reaching Bob, then he will see identical messages from Alice and Eve.) In this attack, Eve cannot make Bob think Alice said something she did not. Eve cannot forge Alice's authentication.

Eve can obviously authenticate M is sent in the clear, because she can just authenticate M with her own public key E. In other words, the attack is only meaningful if M is somehow encrypted. Suppose that M was encrypted as $C = \mathrm{ENC}_k(M)$. If M is encrypted and authenticated from Eve, that Eve must know the message and the message from be her.

Eve can launch the unknown key-share attack on the simplified scheme as follows. She computes $Y = S^{-1}(A(Z))$. Then she computes $Z' = E(Z)$, where E is her public trapdoor function. She replaces (C, T, Z) by (C, T, Z'). When Bob receives the modified message, it will appear to be from Eve. In a sense the message is from Eve, because Eve authenticated it, she just doesn't know what the message is.

Inclusion of the identifiers of Alice and Bob in the key derivation function seems to prevent this attack. (Resisting unknown key share attacks is not primary objective of deniable authentication, so this property is not investigate in greater detail here.) Alternatively, Alice could encrypt Z as well as M, which also seems to prevent the attack because Eve seems to need Z to get Y.

3.2.1 Key Compromise Impersonation

If Eve obtains Bob's private key b, she can impersonate anybody to Bob, including Alice. This is called key compromise impersonation in the context of key agreement schemes. Key compromise impersonation is true for any single-pass deniable authentication scheme, because Bob is capable of producing validly authenticated messages, and thus so is Eve.

3.2.2 Higher Iteration Variants

The scheme uses an alternating application of Alice and Bob's trapdoor and trapdoor inverse functions, respectively. The scheme can be varied by applying these functions more often, such as:

$$Z = f(\dots B(a(B(x)))\dots) \tag{1}$$

where the outermost function f is a or B depending on whether the number of layers is even or odd. (The intermediate bijections have been omitted for simplicity.)

These variants have more costly performance, but some may provide some different kinds of security, as discussed later. (Some variants are completely insecure, such as $Z = B(a(x))$.)

4 Security Analysis of the RSA Scheme

First, the simple proof of deniability is presented. Then some proofs of unforgeability (authenticity) are sketched.

4.1 Deniability

To demonstrate the deniability of the scheme, we need to show that Bob could have generate the pair (Z, x) without Alice's help. Bob can do this just by choosing random $Z \in [A]$ and then computing $x = b(A(Z))$, repeating as necessary until $Y = A(Z) \in [b]$.

The pairs (Z, x) generated by Alice and by Bob are indistinguishable. In both cases, the intermediate value $Y = A(Z) = B(x)$ is uniformly distributed on the set $[a] \cap [b]$. Therefore Bob cannot prove that any pair (Z, x) was created by Alice, provided x was chosen randomly by Alice. So the pair (Z, x) created by Alice is not evidence in itself of Alice's involvement in the key transport scheme.

The scheme does not have *forward deniability*, in the sense of [dRG05], because Alice can choose x non-randomly to enable her to later prove that she generated x and not Bob. Forward deniability, however, is a slight misnomer in that if Alice's system is compromised, then the deniability of her past messages is lost. (Compare to *forward secrecy*: if Alice's system is compromised, then the secrecy of her past messages is retained.) A better term for forward deniability is retractable deniability.

4.2 Security From Trapdoors

The standard RSA assumption is that the RSA is a trapdoor function. Under this assumption we can prove that the RSA scheme is unforgeable, provided the key derivation function and intermediate bijection is modeled as a random oracle.

Theorem 1 (Unforgeability from a Trapdoor). *If the key derivation function H is a random oracle, the bijection S is a random oracle, and an adversary Eve can forge a message (M, T, Z) that Bob thinks is from Alice, then Eve can compute either Alice's private functions a or Bob's b, or Eve can forge a MAC tag with an unknown random key.*

Proof. Suppose that Eve's forgery is (M, T, Z). Because Bob accepts this from Alice, he must compute a value $x = b(S^{-1}(A(Z)))$, a value $k = H(x, A, B, T)$, and he will find that $T = \mathrm{MAC}_k(M)$ holds.

Before generating this forgery, Eve makes queries to the random oracle function H and the random oracle bijection S and its inverse S^{-1}. Let x_i be the value corresponding x in the i^{th} query to H and let k_i be the output. Let (A_j, B_j) correspond to the j^{th} query to S or S^{-1}, such that $A_j = S(B_j)$. A query to S has input B_j and output A_j, and a query to S^{-1}, vice versa.

If $x \neq x_i$ for all i, then $k \neq k_i$ with overwhelming probability. Eve will not know k, and thus her production of T is a MAC forgery. More formally, when $x \neq x_i$ for all x_i, then instead of computing k as random output of H, we instead invoke a MAC function with an unknown random key. Eve has forged this MAC even though we did not know the key.

The remaining possibility is that $x = x_i$ for some i. In this case,

$$B(x_i) = B(x) = S(A(Z)). \tag{2}$$

If $(A(Z), B(x_i)) \neq (A_j, B_j)$ for all j, then in the computation of x the oracle pair computes $S^{-1}(A(Z))$ for the first time and give a random output R. The probability that (2) holds, which is that $R = B(x_i)$, is then negligible. Therefore, we can conclude $(A(Z), B(x_i) = (A_j, B_j)$ for some j, with overwhelming probability. We can assume, without loss of generality, that j is the smallest value for which this is true. Since this is the first time this input-output pair has been seen, the output, whichever of A_j or B_j that is, will be random.

If the j^{th} query is to S, then its input is $B_j = B(x_i)$. Its output is some random value R, because this is the first time input-output pair has been seen. But Eve eventually finds Z such that $A(Z) = A_j = R$. Therefore Eve has inverted A on random input.

If the j^{th} is to S^{-1}, then its input is $A_j = A(Z)$. Its output is some random value R, because this is the first time the input-output pair has been seen. But Eve eventually finds x_i such that $B(x_i) = B_j = R$. Therefore Eve has inverted B on random input. □

This reduction is actually efficient, because of the following trick. If one wants to invert A on random R. Then we can output a value $A_j = R_j = RA(U_j)$. When Eve inverts A on R_j to get Z, we have Z/U_j as the inverse of R. Similarly reasoning applies to inversion of B.

Strictly speaking the proof above only addresses *passive adversaries*. An active adversary Eve can ask Alice to authenticate chosen messages to Bob. Eve can also ask Bob to verify chosen tags. The reduction above must be slightly modified so that Eve's queries to Alice and Bob can be answered even though their private keys are not known to the reduction.

When Eve asks Alice to authenticate message M_k, we will make a virtual query to the S oracle. We choose random (Z_k, x_k) and set $(A_k, B_k) = (A(Z_k), B(x_k))$. Neither A_k nor B_k is the input or the output. Each is simultaneously both input and output. Both values are random, so Eve will still regard S as a random oracle. The rest of the argument holds.

Queries to Bob are handled more easily. A query (M_l, T_l, Z_l) to Bob is rejected if $A(Z_l)$ has not yet been in an input-output pair for S or its inverse. If it has appeared as such a pair and it is not the output of Alice's query, then it is either invalid or a forgery. Therefore we reject all queries to Bob that are not outputs from Alice. This is the correct response, one that Eve will think is real, unless the query is a forgery. If it is a forgery, then we use the previous arguments to break the trapdoor functions or the MAC.

4.3 Security From Claw Resistance

The assumption of a random oracle oracle bijection can be removed, if a stronger assumption about the trapdoor functions is made. This proof applies even if the bijection is the trivial identity function, provided the stronger condition holds.

4.3.1 Claw Resistance

A *claw* for a pair of functions (F, G) is a pair (u, v) such that $F(u) = G(v)$. The pair (F, G) is *claw resistant* if it is not easy to find a claw for the pair. The basic deniable authentication scheme, without the intermediate bijection, relies the claw resistance of the trapdoor functions.

The following informal conjecture about the claw resistance of RSA functions is plausible, if not reasonable.

Conjecture 1 (RSA Claw Resistance). *Let A and B be two different RSA trapdoor functions. The pair (A, B) is claw resistant, unless the associated exponents are very small.*

As already noted earlier, if A and B share a small common exponent e, then a claw can be found as follows. Choose a $Z < \sqrt[e]{A}, \sqrt[e]{B}$. Then (Z, Z) is claw[1] because $A(Z) = Z^e = B(Z)$. More generally, if the exponents are a and b, respectively, and they divide a common small value e, then one has a claw $(Z^{(e/a)}, Z^{(e/b)})$.

(If an adversary with access to a claw generating oracle cannot find an additional claw, then the pair of functions is *adaptively claw resistant*. The adaptive claw resistance of RSA also seems a reasonable assumption.)

[1] An anonymous referee duly informed of this clever attack.

When S is not the trivial identity function, we are interested in the claw resistance of the pair $(A, S \circ B)$.

4.3.2 Security Result

Again, we rely on the key derivation function being a random oracle. The bijection S can be anything however, provided claw resistance is achieved with A and B.

Theorem 2. *If the key derivation function* KDF *is a random oracle hash H and Eve can forge a message from Alice to Bob, then Eve can break the MAC or can find a claw for $(A, S \circ B)$.*

Proof. Suppose Eve's forgery is (M, T, Z). To verify, Bob would compute $x = b(S^{-1}(A(Z)))$, then $k = H(x, A, B, T)$ and then he would get $T = \mathrm{MAC}_k(M)$. Note that $A(Z) = S(B(x))$, so (Z, x) is a claw for $(A, S \circ B)$.

Let x_i be the input corresponding to x in the i^{th} query to H. If $x \neq x_i$ for all i, then the value of k is a random value that Eve has never seen, so Eve must be able to break the MAC function.

Otherwise $x = x_i$, where, without loss of generality, i is the smallest of such values. Now, since x_i is the input to an H query, Eve has found x_i. Eve eventually finds Z, and $(Z, x_i) = (Z, x_i)$, which is a claw, so Eve has found a claw. □

Again, the proof above works for *passive adversaries*. To address active adversaries, we must answer Eve's queries to Alice and Bob as well.

A query to Alice is handled by choosing a random k_j and Z_j, then computing the corresponding T_j. The corresponding value of x_j is not known because we do not have Bob's private key, but we can check if $x_i = x_j$, because $S(B(x)) = S(B(x_i)) = A(Z_j)$. So, when Eve sends such a hash query, we can output $k_i = k_j$ and be consistent with Alice's output. The rest of the argument then applies.

Queries to Bob are handled in the same way as before.

4.4 Security of Other Iterated Variants

We briefly consider using different patterns of alternating compositions of the a and B functions. For simplicity, we assume that S is the trivial identity bijection.

- Suppose that $Z = B(a(x))$. This is insecure because an (a, b)-claw $(Z, x) = (B(y), A(y))$ for any y makes impersonation of Alice to Bob possible.
- Suppose that $Z = a(B(a(x)))$. If Eve can find an (A, B)-claw (Z, x'), then she has $Z = a(B(a(A(x'))))$, so she can impersonate Alice to Bob. Thus this variant is cannot be much more secure than $Z = a(B(x))$.
- Suppose that $Z = B(a(B(x)))$. If Eve can find an (A, B)-claw (Z', x), then she has $Z = B(Z') = B(a(B(x)))$, so she impersonate Alice to Bob. Thus this variant is cannot be much more secure than $Z = a(B(x))$.
- Suppose that $Z = B(a(B(a(x))))$. If Eve can find an (A, B)-claw (Z', x'), so that $Z' = a(B(x'))$, then she can find $(Z, x) = (B(Z'), A(x'))$ such that $Z = B(Z') = B(a(B(x'))) = B(a(B(a(x))))$, enabling her to impersonate Alice to Bob. Thus this variant is cannot be much more secure than $Z = a(B(x))$.

- Suppose that $Z = a(B(a(B(x))))$. If Eve can find two (A, B)-claws (Z, x') and (Z', x) such that $x' = Z'$, then she forge (Z, x). The security of this system is thus based on a problem that might be harder than the (A, B)-claw problem. In fact, finding a (Z, x) is equivalent to the $(b \circ A, a \circ B)$-claw problem.

Roughly speaking, an initial B or terminal a may be dropped from a security perspective. Using more iterations increases the performance cost, while it does not seem to appreciably increase the basic underlying security.

5 Security of Multicasted Deniable Authentication

Suppose that Alice wishes to authenticate a long message to multiple recipients. The obvious approach is to establish a key with each recipient and then to compute a separate MAC tag on the message for each message. The cost of this approach is proportional to:

$$\text{(Length of message)} \times \text{(Number of recipients)}$$

Therefore, if the number of recipients is large and the message is long, this approach can grow very expensive. In these situations, Alice would much prefer a cost which is proportional to a sum of the two factors above.

Note that this section applies to any deniable authentication scheme, including ones based on Diffie-Hellman or ECMQV. Some details may be illustrated with the RSA-based scheme for convenience.

5.1 Key Wrapping

To avoid repeated MAC computation, Alice can use a single common MAC key c for all recipients. She then needs to securely transport c to each of the recipients.

One way to authenticly transport c to multiple recipients, and the approach taken in S/MIME [Hou99], is as follows. Alice agrees on distinct keys with each recipient. Say that key k_B is agreed with Bob and k_C, with Charles. Then Alice *wraps* the key c for each recipient using that recipient's agreed key.

(Wrapping is essentially encryption.) Alice wraps c with key k_B to obtain a wrapped key w_B for recipient Bob. Bob unwraps w_B using k_B to obtain c. Now Bob can verify the MAC tag $T = \text{MAC}_c(M)$ of the message.

5.2 A Multicasting Attack

The following attack is well-known, and has been noted in [BWBL02], for example.

A malicious recipient Charles can unwrap w_C with k_C to obtain c. Now that Charles has the key c, he can abuse if for an attack. He can pick any message M' and compute $T' = \text{MAC}_c(M')$. Charles can send w_B, T' and M' to Bob, along with any other key establishment data that Alice send to Bob to agree on k_B. When Bob receives this, he may believe that M' came from Alice.

If Charles can intercept the original message, then it can replace it with M'. If not, Bob will receive two messages M and M', a genuine one from Alice and a forged one from Charles.

5.3 Countermeasure: Hashing the Tag

If Alice does not trust Charles, then she needs a countermeasure to the multicasting attack. Alice can include the tag T in the input to the key derivation. This seems to stop Charles from launching the attack.

Intuitively, the reasoning for this is as follows. If Charles changes T to T' this will cause Bob to get a different value k'_B for k_B, or a different value w'_B for w_B, respectively. In either case, Bob will get a different value c' for c, a value which Charles cannot determine. Therefore, Charles will not be able to produce the correct tag.

If we assume that the key derivation is random oracle, then we can prove this result more rigorously.

Theorem 3. *If the key derivation function used to compute recipient keys k is a random oracle hash H, the key wrap function is such that unwrapping W with a random key k gives indistinguishable from random result c, the MAC function is unforgeable and collision-resistant, the input to the hash includes the MAC tag T, and the underlying key agreement scheme is secure in the sense Bob has a shared secret x with Alice that nobody else can compute, then Charles cannot impersonate Alice to Bob.*

Proof. Suppose that Charles produces a forgery (M, T, W, Z) where Alice did not authenticate message M. Here, Z is some key agreement data that Bob receives purportedly from Alice, so that we interpret the notation of the RSA scheme more generally. The values T and W are MAC tag and wrapped key W.

Bob will first compute his shared secret key as $x = F(b, A, Z)$ where F is a key agreement function, b is Bob's private key, and A is Alice's public key. Then, Bob will derive his recipient key as $k = H(x, T)$. Next, Bob recovers $c = \text{Wrap}^{-1}(k, W)$. Finally, Bob finds that $T = \text{MAC}_c(M)$.

Let (x_i, T_i) be the i^{th} query input to the H oracle. If $(x, T) \neq (x_i, T_i)$ for all i, then $k = H(x, T)$ has some random value. This means that c is random value, by nature of the wrap function. In this case, Charles can break the MAC.

Otherwise $(x, T) = (x_i, T_i)$ for some i, which we will again assume is the smallest such i. Before making this query, Charles knew x and T. Again, the output $k = k_i$ is random, so the c can be argued to be random. Thus Charles can break the MAC.

The above argument only works for passive adversaries, so does not say much. We now presume that Charles has access to an Alice oracle, which authenticates a message of Charles' choice to both Bob and Charles.

The input to the Alice oracle is a message M_j. The output is $(T_j, W_j, Z_j, W'_j, Z'_j)$ where W'_j and Z'_j are intended for Charles. Charles can use this values to derive his recipient key and unwrap W'_j to obtain the MAC key c_j. Charles can compute T_j from c_j, so he does not really need it. For convenience, we simplify the Alice oracle by regarding its output as (W_j, Z_j, c_j).

Now $k_j = H(x_j, T_j)$, where k_j is Bob's recipient key for M_j and the x_j is his raw agreed key. We assumed that Charles cannot learn x_j. We regard (x_j, T_j) as an input to the hash oracle, but one whose input Charles does not get to choose the input. Charles learns one of the inputs, T_j but not necessarily the output k_j or the other input x_j. The same argument as before still applies, however, that (x, T) in the forgery must be one of the hash queries, or else Charles can break the MAC.

Again as before, since Charles cannot have broken the MAC, we can also deduce $(x, T) \neq (x_i, T_i)$ for his own hash queries (otherwise k_i is random and Charles can break the MAC).

Therefore, Charles must have somehow arranged that $(x, T) = (x_j, T_j)$ from one of the queries to Alice. In this case, $k = H(x, T) = H(x_j, T_j) = k_j$.

If $W = W_j$, then $c = c_j$ because the unwrap function is deterministic. If $M \neq M_j$ then Charles has found a collision in the MAC function, because $\mathrm{MAC}_c(M) = T = T_j = \mathrm{MAC}_c(M_j)$. Of course if $M = M_j$, then M is not a forgery. This is a message-collision in the MAC.

If $W \neq W_j$, then certainly $c \neq c_j$. Now Charles has found a collision in the MAC again, because $\mathrm{MAC}_c(M) = T = T_j = \mathrm{MAC}_{c_j}(M_j)$. This is a key-collision in the MAC. □

For MAC such as HMAC, a MAC collision gives a collision in the underlying hash. Hash functions are designed to a level of collision resistance, however this may not be as high as their other security properties.

A significant improvement in the result above would be not too rely on collision resistance of the MAC function. However, it is not clear that one can rely on a purely standard assumption about the MAC. For example, if Charles can choose a message M, get a MAC key c and find a MAC message-collision M' with M under key c, then he can still impersonate Bob. This may be possible even with an unforgeable MAC.

5.4 Other Countermeasures

Some alternatives to hashing the tag T in the key derivation function are:

- Include T as an optional authenticated input to the key wrap function. The security of this approach depends very much on the properties of the wrap functions. The existing wrap functions are not yet very well-studied.
- Compute a separate MAC tag for each recipient, but do so on a digest of the message, either as a hash or as another MAC tag. Tagging a tag is preferable to tagging a hash from a security standpoint, because then one does not need to rely on collision resistance of the hash function. On the other hand, hashing has the advantage the message can be hashed independently for the key establishment operations.

5.5 Specific Application to the RSA Scheme

The RSA scheme can support multicasting by using a key wrap algorithm and the methods above, or multicasting can integrated more deeply into the scheme. In the integrated form, Alice chooses a value x such that $B(x)$ belongs to all of the domains of the recipients. If sender and recipient domains are similar in size, this will generally only take one try at a random x. The key K derived from x incorporates the tag T, so Charles cannot tamper with the message to impersonate Alice to Bob.

5.6 Replay Attacks

In a replay attack, Eve resends an old authenticated message (M, T, Z) sent by Alice to Bob. Eve gets Bob to think she is sending a message again, when she is not. Strictly speaking, replay resistance is not a formal goal of deniable authentication[2]. The primary goal is to resist forgery, where Eve makes Bob accept a message that Alice never sent. Nevertheless,

[2]Replay resistance is not usually required for MAC schemes, so why should it be require for its asymmetric variant.

it is reasonable to have a countermeasure against replay attacks. Resisting replay attacks can be done with techniques similar to those for resisting multicasting attacks.

The time (or just the date) can be included in the message. If Alice and Bob have roughly synchronized clocks, then Bob can accept the message if its time is within the allowed tolerance from message transmission. For potentially stronger protection, the time can put into the key derivation function or key wrap function.

More generally, other increasing nonces can be used. (Time is essentially special kind of nonce.) These require, however, that Alice and Bob keep track of the latest nonce. An advantage of nonces is that they are not limited to a smallest division of time, such as a second or day. A disadvantage of strictly increasing nonces is that the messages arrive out of order, then some valid messages may be rejected, unless some tolerance is admitted.

6 Conclusion

Notwithstanding the results in this paper, it is too early to recommend deniable authentication with RSA, if more well-studied approaches based on DH or ECMQV are available. No compelling advantage of RSA over alternatives exists. Nevertheless, if deniable authentication is required and only RSA public keys are available, then this scheme may be used, and thus may be appropriate for inclusion in standards for RSA. Mostly, it is interesting to note that deniable authentication with RSA is possible[3] at all, given that the usual padding schemes for securing RSA make it impossible.

If deniable authentication proves important in the future, then presumably securing its multicast form will also prove important. It is important to avoid the pitfalls mentioned in this paper. One of the countermeasures described here may become useful.

References

[BWBL02] S. Blake-Wilson, D. Brown, and P. Lambert, *RFC 3278: Use of ECC algorithms in CMS*, April 2002, `www.ietf.org/rfc/rfc3278.txt`.

[CS03] J. Camenisch and V. Shoup, *Practical verifiable encryption and decryption of discrete logarithms*, Advances of Cryptology — CRYPTO 2003 (D. Boneh, ed.), LNCS, no. 2729, IACR, Springer, August 2003, `shoup.net/papers`.

[Dav01] D. Davis, *Defective sign & encrypt in S/MIME, PKCS # 7, MOSS, PEM, PGP and XML*, `theworld.com/~dtd/sign_encrypt/sign_encrypt7.html`, May 2001.

[dRG05] M. di Raimondo and R. Gennaro, *New approaches for deniable authentication*, ePrint 2005–46, IACR, February 2005, `eprint.iacr.org/2005/046`.

[Hou99] R. Housley, *RFC 2630: Cryptographic message syntax*, June 1999, `www.ietf.org/rfc/rfc2630.txt`.

[MvOV95] A. J. Menezes, P. van Oorschot, and S. A. Vanstone, *The handbook of applied cryptography*, CRC Press, 1995.

[3] For example, even this author once thought that deniable authentication with RSA was fundamentally impossible.

A Deniable Public-Key Encryption

Most public-key encryption algorithms have the property that the decrypter can reveal the decrypted plaintext to third parties without compromising the security of the decryption key. Furthermore, a common side effect of this is that third parties can verify the validity of this decryption. (To do this, the decrypter may need to provide additional information to the third party, such as padding.) In other words, most public-key encryption algorithms are *undeniable*, in the sense that the decrypter can prove to others that correct match between ciphertext and plaintext. An encryption is *deniable* if the decrypter cannot prove to third parties the correspondence between the plaintext and ciphertext. Most symmetric-key encryption algorithms are half-deniable in the sense that proving the match between plaintext and ciphertext requires revealing the private key, so cannot be proved to untrusted third parties.

Given a deniable public-key encryption scheme, signing of a deniably encrypted message is a form of weakly deniable authentication, because something is signed, but the message cannot be bound to the signer. This is similar to encrypting a symmetric authentication key, and may not have any advantages over it.

The deniability of encryption may be a related to the notion of *verifiable encryption* defined by Camenisch and Shoup [CS03].

B System Wide Deniability

In some infrastructures, email is often forwarded hop to hop by several mail servers before reaching its destination. Evidence of this trail, if available, may be hard for Alice to deny. If mail servers sign each hop, then deniability is quite hindered. Ultimately, if full deniability is to be achieved at a system level, signing message hops should be avoided.