An Attack on Server Assisted Authentication Protocols

Indexing term: Information theory

The basic server-assisted authentication protocol of Matsumoto, Kato and Imai can be broken in a one-round active attack. The improvements necessary to make it secure may well render it impractical.

Introduction: Matsumoto, Kato and Imai proposed various protocols in [1] to speed up secret computations using insecure auxiliary devices. A typical application would be where a smart card wishes to calculate an RSA signature \( m^d \pmod{n} \) [2] on a financial transaction and wants computational assistance from a powerful server such as a digital signal processor located in a point of sale device. However, the cardholder would not wish to trust this server with \( d \) because of the risk of false terminal attacks.

This problem had been raised by Feigenbaum in [3], and the proposed solution is intended for use in production systems [4]. This would be unwise, as the server can almost trivially determine the card’s secret key.

Proposed protocol: In the simplest version of the protocol, the smart card wishes to sign a message \( m \) with a secret RSA key \( d \), that is, to calculate \( m^d \pmod{n} \). To do this, it generates integers \( d_i \) and binary weights \( w_i \) such that

\[
d = \sum_{i=0}^{k} d_i w_i \pmod{\phi(n)}
\]  

(1)

It now sends \( n \), the vector \( d_i \) and the message \( m \) to the server, which calculates

\[
z_i = m^{d_i} \pmod{n}
\]  

(2)

and returns the \( z_i \). The card can now calculate \( m^d \) by multiplying together those \( z_i \) for which \( w_i = 1 \).

Attacking the protocol: Shimbo and Kawamura devised an active attack on a derived protocol [5], while various multi-round active and passive attacks
are discussed by Burns and Mitchell [6] and Pfitzmann and Waidner [7]. In this
letter, we show a simple one-round attack on the basic protocol which reveals
the card’s secret key.

Instead of returning the $z_i$ as in equation (2) above, the server chooses a
random number $r$ and returns $z_i = p_i r$ where the $p_i$ are random primes chosen
so that their product is less than $n$. Now when it receives $\prod_{w_i=1}^{\Pi} z_i$ back from
the card, the server repeatedly divides this number by $r$ until the result is
$\prod_{w_i=1}^{\Pi} p_i$, which it can factor, thus obtaining $w_i$ and hence $d$.

The server can now calculate the correct signature $m^d$ for the message and
forward the transaction to the network. The sale will proceed normally and so
the customer has no indication that his secret key has in fact been compromised.

**Can the protocol be fixed?** As suggested in [5], [6] and [7], active attacks
may be prevented if the card checks the signature $m^d$ before sending it back
to the server. Our attack makes such a check imperative, and this reduces the
computational advantage to be gained from a protocol of this type, even when
the checking is carried out using a relatively sparse public modulus.

Furthermore, as observed in [6] and [7], if the card returns a signature only
when it checks, then this makes active attacks possible in which the server
determines the nonzero values of $w_i$ by trial and error. Thus the card would
have to disable itself after an error was detected, or choose a new set of $d_i$ as
part of the precomputation.

It also seems likely that if the card has the storage to deal with $d_i$ and $z_i$,
then it might just as well be able to speed the calculation of a DSA signature by
standard precomputation techniques [8]. In any case, full RSA signatures can
now be calculated on smartcards in under half a second for a 512 bit modulus
[9].

Server assisted protocols are therefore no longer needed in the point-of-sale
environment for which they were invented, and the added complexity needed
to make them secure against active attacks will probably ensure that they offer
at best a marginal performance benefit, especially when the communication
overheads are taken into account. It therefore quite unclear that these protocols
are of any practical use.

R J ANDERSON

*University Computer Laboratory*

*Pembroke Street, Cambridge CB2 3QG*
References


