

Re: ADVERT: Secure comms

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2006-08/msg01677.html>

- *From:* Mark Wooding <mdw@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 30 Aug 2006 23:57:53 +0000 (UTC)
-

Robin Carey <robin_carey5@xxxxxxxxxxxx> wrote:

C12-GAMMA is a finalised branch of Caesarion v12; a free software product for the FreeBSD and Linux operating systems.

Hmm. It smells like snake-oil, but it's free software. This is strange.

Quoting from the website:

: The use of the L15 Stream Cipher which is a technological superiority
: in itself.

This line just makes me nervous. Still...

: Innovative Cipher-Packet technique hides ciphertext in padding thus
: preventing cryptanalysis.

Huh? Are you referring to the absurd way you pack plaintext into
'padded' integers prior to RSA encryption?

: E-mail is encrypted using the RSA public-key cryptosystem thus
: eliminating security risks from symmetric ciphers.

Uhhh... What would they be?

: The provision of secret-public-key facilities caters for high security
: scenarios.

You've lost me. Why on earth would I want to prat about with RSA when I
don't need public key crypto? I'd reach for AES.

: Unorthodox (reversed) RSA encoding of data should provide a higher
: level of security relative to orthodox implementations.

This is your crazed padding scheme again, isn't it? I'll discuss that
below.

Re: ADVERT: Secure comms

: Digital signatures are encrypted thus eliminating security risks from
: cryptographic hash functions (most of which have recently discovered
: security issues).

FUD. SHA1's pseudorandomness properties haven't been attacked.

Of course, this doesn't stop you from using SHA1 to hash your messages as part of your digital signature scheme, and SHA1's collision resistance — a necessary property for digital signatures — has indeed been attacked.

: It is distributed under a software license which mandates that it may
: not be used for terrorism, paedophilia or crimes against humanity.

.... because we know that terrorists and war criminals abide by their licence agreements.

Also, that clause technically stops your program from being free software according to the FSF definition (it fails to satisfy freedom 0, 'the freedom to run the program, for any purpose'), or the Debian Free Software Guidelines (contradicts 5, 'No Discrimination Against Fields of Endeavor'), and it fails to meet the Open Source Definition (for the same reason as it fails the DFSG).

On to the code...

```
// NOTE: Ideally, saltLen should be in the range 0 -> 255. However, the  
// BN_bin2bn() function in the OpenSSL BIGNUM library renders the  
// same BIGNUM value for the two distinct binary representations of:  
// { 0, 1, 2, 3 } and { 1, 2, 3 }. This is a bug as far as I'm  
// concerned. Thus, saltLen has to be in the range 1 -> 255.
```

Duh! What different value were you expecting? If the function were to compute something other than the integer represented in the given byte array in radix-256 (using some byte-ordering) I'd call that a bug. Since it computes that correctly, it seems you're expecting something strange of it.

So, the actual encryption scheme...

The message is split into blocks. The length of each block is chosen at random. The maximum block length is half the size of the recipient's public key, rounded down; call this value M. The block size is chosen between 0 and M, inclusive.

Each block m_i is padded by choosing a random length n between 1 and M, generating a string r of n random bytes, and forming the padded block $n || r || m_i$. This is then converted to an integer in the obvious big-endian way, and the RSA public-key operation applied to it.

Re: ADVERT: Secure comms

Re: ADVERT: Secure comms

A signature is computed as follows. Let h be the SHA1 hash of the plaintext. Form $p \parallel h \parallel r'$, where p is a random nonzero byte, and r' is a random string whose length is chosen randomly between zero and the maximum size that will fit; this padded string then has the RSA private-key operation applied, and the result is split into blocks and encrypted in the same way as the original message.

Blocks are output in hexadecimal; blocks representing encrypted signature data begin with a `S'; message ciphertext blocks are not prefixed.

This is all, of course, completely mad. The block padding scheme entirely fails even to approximate a uniform distribution on the domain of the RSA function, so you can't claim security based on the standard RSA problem. Indeed, with probability at least 2^{-16} , a padded message block consists of precisely three bytes: a 1 byte, a random byte, and a message byte. Thus, a $O(2^{16})$ search will successfully decrypt one block in at least 65536, which is obviously unacceptable.

The signature scheme looks like it's probably forgeable, using the techniques used to attack PKCS#1 1.5 signature padding, only the attack looks easier.

Of course, using RSA blockwise to encrypt a message is very slow, and decrypting is even slower; but the decryption program doesn't even use the (extremely well-known, completely standard) Chinese-Remainder-Theorem optimization for doing RSA private-key operations, so its performance sucks mightily. This is particularly amusing, since the program uses libssl for its crypto, which already implements this optimization.

I'm sorry, but this is just dreadful. I feel sort of dirty that I spent my time staring at this travesty. Certainly it's a stain on the security reputation of free software.

— [mdw]

.