

Re: OTP and message integrity.

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2003-06/0854.html>

From: Mark Wooding (mdw_at_nsict.org)

Date: 06/19/03

Date: 19 Jun 2003 08:54:24 GMT

contact <contact@eversealsolutions.com> wrote:

> *And why would that be? Without the hash, the integrity couldn't be
> checked at all. Without the keys, the message couldn't be decoded.*

Yes, but there are better ways. You provide some level of integrity, but it's a lash-up rather than a well thought-out scheme. The right way to provide integrity is with a MAC.

The encryption is provided by an OTP. We assume that the random data for the OTP is indeed random, never used before, never used after, secret, known only to the sender and recipient, yes? So XORing the message with the OTP is sufficient for IND-CPA (i.e., secrecy against chosen plaintext attacks).

The whole merry dance with 3DES and SHA1 is therefore either there for show, or to provide integrity assurances which the OTP can't. I believe the latter to be true.

Note that both parts are necessary for integrity. Suppose, for example, we just had the hash. Suppose, too, that the scheme was being used for military purposes (just so that I can use classic messages), and as a result of espionage, it was known that a particular ciphertext C represented the message $M = \text{'ATTACK AT DAWN'}$. There's some OTP, P , say, such that

$$C = (M \parallel H(M)) \text{ XOR } P$$

(here, \parallel denotes concatenation). Note that we don't know P , though we do know M and we know how to compute $H(\cdot)$.

Let $M' = \text{'HOLD POSITIONS'}$. Note that this has the same length as M . Then we can construct

$$\begin{aligned} C' &= C \text{ XOR } ((M \text{ XOR } M') \parallel (H(M) \text{ XOR } H(M'))) \\ &= C \text{ XOR } (M \parallel H(M)) \text{ XOR } (M' \text{ XOR } H(M')) \\ &= (M \parallel H(M)) \text{ XOR } P \text{ XOR } (M \parallel H(M)) \text{ XOR } (M' \text{ XOR } H(M')) \\ &= (M' \text{ XOR } H(M')) \text{ XOR } P \end{aligned}$$

sci.crypt: Re: OTP and message integrity.

which is a correct encryption of M' . The result is a military catastrophe.

The encryption means an adversary doesn't know the right things to XOR in to change the message. This is what holds the scheme. And compared to a MAC, it's a lash-up.

You could instead use a variant of UMAC (by Black, Halevi, Krawczyk, Krovetz and Rogaway) -- UHASH the OTP ciphertext (using yet more random data you have lying around -- though you can reuse this key material for further messages to the same recipient), and XOR the result with the next few bytes from the OTP file you encrypted the data with. For a 128-bit MAC tag -- 16 bytes of your random data files used up each time -- This will give you a /guaranteed/ mathematically proven forgery probability no larger than 2^{-120} \$. There's nothing an adversary can break -- no unproven assumptions, nothing -- to do better than that.

> *Gee thanks. Still, you're offensive. The hash is a standard SHA hash
> using 160/512 bit primes as per the standard.*

SHA (any version) doesn't use primes. But I'm not complaining that the hash is a bad one -- I'm complaining that a hash function isn't the right tool to provide integrity. That I can't, off the top of my head at midnight or having just woken up, find attacks doesn't mean that it's secure.

> > *an adversary who knows the plaintext*
>
> *If that were so, then every thing else would sorta be irrelevant. With
> the exception of the ability to send a false message. But then the
> attacker would also have to have access to the random number files and
> the code. If he has that, then it seems he has the person's computer
> in it's entirety.*

Not true. In many cases it might be possible to know the plaintext of a message without knowing the keys. Maybe a user always sends the same message every Tuesday lunchtime. Maybe he's forwarding a message he received from the adversary (but doesn't know this). Maybe he's a computer autoresponder and therefore doesn't have much imagination. A secure encryption scheme should work even in these cases.

And is the ability to send a forged message so useless? I think not. See the (silly) military example above.

> *The hash is of the 3DES encrypted data, just before the OTP step.*

Right. That disposes of my attacks. Thanks.

> *BTW the keys are different for each encryption, the random number
> files for the OTP step are different for each encryption.*

Re: OTP and message integrity.

sci.crypt: Re: OTP and message integrity.

Yes, I assumed that that was the case. I also assumed that they were chosen uniformly at random, and were unknown to the adversary.

> > *I can't see any more attacks offhand, though this looks brittle to me.*
> > *Hashing the ciphertext may well be more secure. XORing the hash with*
> > *the OTP makes it look a little like an XOR-universal-hash-based MAC, and*
> > *the remaining difference -- the ability for the adversary to know the*
> > *XOR difference between the hashes of two messages*
>
> *How would that be possible when a different random number file is used*
> *each time for the OTP step?*

It'd be possible because the hash function (SHA1) is public knowledge. It's specified in a well-known government standard, after all. But as I said, the encryption thwarts that.

> > *I stand by my description of EverSeal as snake-oil.*
>
> *Gee thanks again. I doubt I care. I mean after all, this is not 'Virtual*
> *Matrix Encryption' or 'we use a secret encryption system', phrases anyone*
> *can laugh over.*

Whereas OTPs get used in many respectable programs. And crypto systems where the keys are sent to you by someone else, they're real high on my list of good ideas.

> *Which brings up a point -- what are you doing?*

This'n'that. I write free software (in the GNU sense of the words). I maintain a crypto library and I'm slowly adding elliptic curve support. I need to get around to proving the key exchange protocol in my VPN software secure (and maybe get someone to pay for a trip to a crypto conference with the results).

Oh, and I work for a company which makes crypto hardware.

> *The whole point was to provide well formed random number files. Which is*
> *where most OTP schemes fail.*

Errr, yes, so you generate the random data yourself (how?) and send them to the users securely (how?).

> *Do you think a PRNG would be better?*

Actually, the triple-DES encryption looked good enough to me until you stuck the keys on the end. Since you rely on 3DES and SHA1 for security anyway, I'd go with 3DES-CBC encryption and an HMAC-SHA1 MAC; either change the keys regularly (to stay ahead of the CBC collision bounds) or use a public key scheme to hand over new keys each time.

sci.crypt: Re: OTP and message integrity.

> *And maybe I haven't earned it fully but most all of you are using an*
> *OS or browser that you have no idea what it does.*

I've read quite a lot of the source code to my operating system. I've not looked at my web browser but I could if I wanted to.

> *>Oh, dear. The Crypto-Gram Doghouse beckons,*
>
> *Been there, done that.*

Oh. I thought you might have been (fitted the profile, name rang a bell) so I grepped my archive and couldn't find you.

> *It's a shame that the man feels he has to attack others to make*
> *himself feel good.*

No, that's not what the Doghouse is about. It's about warning people away from bad cryptography — teaching them about the warning signs and alerting them to specific examples.

Thanks for this conversation. You're obviously intelligent, articulate and pleasant, which counts for a lot. Which leaves me wondering /why/ you're peddling snake-oil. Maybe if I can get through to you what the differences are, you'd renounce the dark side and make a valuable contribution to `real' crypto.

Take a look at UMAC, anyway.

<http://www.cs.ucdavis.edu/~rogaway/umac/index.html>

-- [mdw]