A Few Thoughts on Cryptographic Engineering

Some random thoughts about crypto. Notes from a course I teach. Pictures of my dachshunds.

Saturday, May 19, 2012

How to choose an Authenticated Encryption mode

(source/cc)

If you've hung around this blog for a while, you probably know how much I like to complain. (Really, quite a lot.) You might even be familiar with one of my favorite complaints: *dumb crypto standards.* More specifically: dumb standards promulgated by smart people.

The people in question almost always have justifications for whatever earth-shakingly stupid decision they're about to take. Usually it's something like '*doing it right would be hard*', or '*implementers wouldn't be happy if we did it right*'. Sometimes it's '*well, we give the option to do it right*'. In the worst case they'll tell you: '*if it bothers you so much, why don't you join the committee and suggest that idea yourself, Mr. Smartypants*'.

Well, first of all, it's *Dr.* Smartypants. And moreover, I've tried. It doesn't work.

Case in point: I happen to be lurking on the mailing list of a standards committee that recently decided to allow unauthenticated CBC mode encryption as an option in their new web encryption standard. When I pointed out that *the exact same decision* led to the failure of a previous standard -- ironically, one that this new standard will probably replace -- I was told, politely, that:

1. Mandating authenticated encryption would be hard.
2. Real implementers don't know how to implement it.
3. We already offer the *option* to use authenticated encryption.
4. Stop telling us things we already know.

The worst part: they really *did* know. The committee included some smart, smart people. People who know that this is a bad idea, and who have decided either to just *go with it*, or else have convinced themselves that implementers won't *(a)* pick the easy, insecure option, and then *(b)* screw it up completely. I have news for these people: *Yes, they will*. This is why we write standards.

After all this build-up, it may surprise you that this is not a post about standards committees. It's not even a post about smart people screwing things up. What I'm here to talk about today is Authenticated Encryption, what the hell it is, why *you* need it. And finally, (assuming you're good with all that) *which* of the many, many AE schemes should you consider for your application.

First, some background.

What's Authenticated Encryption and why should I care?

For those of you who *don't* know what AE is, I first need to explain one basic fact that *isn't* well explained elsewhere:

> *Nearly all of the symmetric encryption modes you learned about in school, textbooks, and Wikipedia are (potentially) insecure.*

This covers things like AES when used in standard modes of operation like CBC and CTR. It also applies to stream ciphers like RC4. Unfortunately, the list of potentially insecure primitives includes many of the common symmetric encryption schemes that we use in practice.

About Me

Matthew Green

I'm a cryptographer and research professor at Johns Hopkins University. I've designed and analyzed cryptographic systems used in wireless networks, payment systems and digital content protection platforms. In my research I look at the various ways cryptography can be used to promote user privacy.

My website
My twitter feed
Useful crypto resources
RSS
Bitcoin tipjar
Matasano challenges

Journal of Cryptographic Engineering
(not related to this blog)

View my complete profile

Popular Posts


On the NSA
Let me tell you the story of my tiny brush with the biggest crypto story of the year . A few weeks ago I received a call from a reporter a...


What's the matter with PGP?
Image source: @bcrypt . Last Thursday, Yahoo announced their plans to support end-to-end encryption using a fork of Google's end-to-...


Attack of the week: FREAK (or 'factoring the NSA for fun and profit')
Cryptography used to be considered 'munitions'. This is the story of how a handful of cryptographers 'hacked' the NSA....


Here come the encryption apps!
It seems like these days I can't eat breakfast without reading about some new encryption app that will (supposedly) revolutionize our c...


Truecrypt report
A few weeks back I wrote an update on the Truecrypt audit promising that we'd have some concrete results to show you soon. Thanks to so...

Now, I want to be clear. These schemes are *not* insecure because they leak plaintext information to someone who just intercepts a ciphertext. In fact, most modern schemes hold up amazingly well under that scenario, assuming you choose your keys properly and aren't an idiot.

The problem occurs when you use encryption in *online* applications, where an an adversary can intercept, tamper with, and submit ciphertexts to the receiver. If the attacker can launch such attacks, many implementations can fail catastrophically, allowing the attacker to completely decrypt messages.

Sometimes these attacks requires the attacker to see only an error message from the receiver. In other cases all he needs to do is measure *time* it takes for the receiver to acknowledge the submission. This type of attack is known as a chosen ciphertext attack, and by far the most common embodiment is the 'padding oracle attack' discovered in 2002 by Serge Vaudenay. But there are others.

The simplest way to protect yourself against these attacks is to simply MAC your ciphertexts with a secure Message Authentication Code such as HMAC-SHA. If you prefer this route, there are two essential rules:

1. Always compute the MACs on the ciphertext, never on the plaintext.
2. Use two different keys, one for encryption and one for the MAC.

Rule *(1)* prevents chosen-ciphertext attacks on block cipher modes such as CBC, since your decryption process can reject those attacker-tampered ciphertexts before they're even decrypted. Rule *(2)* deals with the possibility that your MAC and cipher will interact in some unpleasant way. It can also help protect you against side-channel attacks.

This approach -- encrypting something, then MACing it -- is not only secure, it's *provably* secure as long as your encryption scheme and MAC have certain properties. Properties that most common schemes do seem to possess.*

Dedicated AE(AD) modes

Unfortunately, the 'generic composition' approach above is not the right answer for everyone. For one thing, it can be a little bit complicated. Moreover, it requires you to implement two different primitives (say, a block cipher and a hash function for HMAC), which can be a hassle. Last, but *not* least, it isn't necessarily the fastest way to get your messages encrypted.

The efficiency issue is particularly important if you're either (a) working on a constrained device like an embedded system, *or* (b) you're working on a fast device, but you just need to encrypt lots of data. This is the case for network encryptors, which have to process data at line speeds -- typically many gigabytes per second!

For all of these reasons, we have specialized block cipher modes of operation called Authenticated Encryption (AE) modes, or sometimes Authenticated Encryption with Associated Data (AEAD). These modes handle both the encryption and the authentication in one go, usually with a single key.

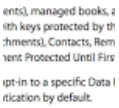
AE(AD) modes were developed as a way to make the problem of authentication 'easy' for implementers. Moreover, some of these modes are lightning fast, or at least allow you to take advantage of *parallelization* to speed things up.

Unfortunately, adoption of AE modes has been a lot slower than one would have hoped for, for a variety of reasons. One of which is: it's hard to find good implementations, and another is that there are tons and tons of AE(AD) schemes.

So, which AE mode should I choose?

And now we get down to brass tacks. There are a plethora of wonderful AE(AD) modes out there, but which one should you use? There are many things to consider. For example:

- How fast is encryption and decryption?
- How complicated is the implementation?
- Are there free implementations out there?
- Is it widely used?
- Can I parallelize it?
- Is it 'on-line', *i.e.,* do I need to know the message length before I start encrypting?



Why can't Apple decrypt your iPhone?

Last week I wrote about Apple's new default encryption policy for iOS 8 . Since that piece was intended for general audiences I mostly ...

Attack of the week: OpenSSL Heartbleed

Ouch. (Logo from heartbleed.com) I start every lecture in my security class by asking the students to give us any interesting security ...

Dear Apple: Please set iMessage free

Normally I avoid complaining about Apple because (a) there are plenty of other people carrying that flag, and (b) I honestly like Apple ...

Let's audit Truecrypt!

[source] A few weeks ago, after learning about the NSA's efforts to undermine encryption software , I wrote a long post urging d...

RSA warns developers not to use RSA products

In today's news of the weird, RSA (a division of EMC) has recommended that developers desist from using the (allegedly) 'backdoore...

My Blog List

Schneier on Security
More on Chris Roberts and Avionics Security
3 hours ago

ellipticnews
Accepted papers at CRYPTO
13 hours ago

Shtetl-Optimized
Five announcements
2 days ago

Bristol Cryptography Blog
52 Things: Number 32: difference between game-based and simulation-based security definitions
4 days ago

Bentham's Gaze | Information Security Research, University College London
Measuring Internet Censorship
5 days ago

root labs rdist
Was the past better than now?
5 months ago

Cryptanalysis
SSLv3 considered to be insucure – How the POODLE attack works in detail
7 months ago

The MPC Lounge
5th Bar-Ilan Winter School 2015: Advances in Practical Multiparty Computation
7 months ago

Chargen

Subscribe

Followers

Posts

- Is it patented?
- Does it allow me to include Associated Data (like a cleartext header)?
- What does Matt Green think about it?

To answer these questions (and particularly the most important final one), let's take a look at a few of the common AE modes that are out there. All of these modes support Associated Data, which means that you can pre-pend an *unencrypted* header to your encrypted message if you want. They all take a single key and some form of *Initialization Vector* (nonce). Beyond that, they're quite different inside.

GCM. Galois Counter Mode has quietly become the most popular AE(AD) mode in the field today, despite the fact that everyone hates it. The popularity is due in part to the fact that GCM is extremely fast, but mostly it's because the mode is patent-free. GCM is 'on-line' and can be parallelized, and (best): recent versions of OpenSSL and Crypto++ provide good implementations, mostly because it's now supported as a TLS ciphersuite. As a side benefit, GCM will occasionally visit your house and fix broken appliances.

Given all these great features, you might ask: why does everyone hate GCM? In truth, the only people who hate GCM are those who've had to *implement* it. You see, GCM is CTR mode encryption with the addition of a Carter-Wegman MAC set in a Galois field. If you just went 'sfjshhuh?', you now understand what I'm talking about. Implementing GCM is a hassle in a way that most other AEADs are *not*. But if you have someone else's implementation -- say OpenSSL's -- it's a perfectly lovely mode.

OCB. In performance terms Offset Codebook Mode blows the pants off of all the other modes I mention in this post. It's 'on-line' and doesn't require any real understanding of Galois fields to implement** -- you can implement the whole thing with a block cipher, some bit manipulation and XOR. If OCB was your kid, he'd play three sports and be on his way to Harvard. You'd brag about him to all your friends.

Unfortunately OCB is *not* your kid. It belongs to Philip Rogaway, who also happens to hold a patent on it. This is no problem if you're developing GPL software (it's free for you), but if you want to use it in a commercial product -- or even license under Apache -- you'll probably have to pay up. As a consequence OCB is used in approximately no industry standards, though you might find it in some commercial products.

EAX. Unlike the other modes in this section, EAX mode doesn't even bother to *stand* for anything. We can guess that E is Encryption and A is Authentication, but X? I'm absolutely convinced that EAX is secure, but I cannot possibly get behind a mode of operation that doesn't have a meaningful acronym.

EAX is a two-pass scheme, which means that encryption and authentication are done in separate operations. This makes it much slower than GCM or OCB, though (unlike CCM) it is 'on-line'. Still, EAX has three things going for it: first, it's patent-free. Second, it's pretty easy to implement. Third, it uses only the Encipher direction of the block cipher, meaning that you could technically fit it into an implementation with a very constrained code size, if that sort of thing floats your boat. I'm sure there are EAX implementations out there; I just don't know of any to recommend.

Whatever you do, be sure not to confuse EAX mode with its dull cousin EAX(prime), which ANSI developed only so it could later be embarrassingly broken.

CCM. Counter Mode with CBC MAC is the 1989 Volvo station wagon of AEAD modes. It'll get you to your destination reliably, just not in a hurry. Like EAX, CCM is also a two-pass scheme. Unfortunately, CCM is *not* 'on-line', which means you have to know the size of your message before you start encrypting it. The redeeming feature of CCM is that it's patent-free. In fact, it was developed and implemented in the 802.11i standard (*instead* of OCB) solely because of IP concerns. You can find an implementation in Crypto++.

The rest. There are a few more modes that almost nobody uses. These include XCBC, IAPM and CWC. I have no idea why the first two haven't taken off, or if they're even secure. CWC is basically a much slower version of GCM mode, so there's no real reason to use it. And of course, there are probably plenty more that I haven't listed. In general: you should use those at your own risk.

Summing up

So where are we?

In general, the decision of which cipher mode to use is *not* something most people make

Blog Archive

► 2015 (7)
► 2014 (13)
► 2013 (23)
▼ 2012 (48)
► December (1)
► November (1)
► October (4)
► September (3)
► August (4)
► July (2)
► June (3)
▼ May (5)
TACK
If wishes were horses then beggars would ride... a...
How to choose an Authenticated Encryption mode
A tale of two patches
The future of electronic currency
► April (6)
► March (4)
► February (7)
► January (8)
► 2011 (39)

every day, but when you *do* make that decision, you need to make the right one. Having read back through the post, I'm pretty sure that the 'right' answer for most people is to use GCM mode and rely on a ~~trusted~~ free implementation, like the one you can get from OpenSSL.

But there are subcases. If you're developing a commercial product, don't care about cross-compatibility, and don't mind paying 'a small one-time fee', OCB is also a pretty good option. Remember: even cryptographers need to eat.

Finally, if you're in the position of developing your own implementation from scratch (not recommended!) and you really don't feel confident with the more complicated schemes, you should serious consider EAX or CCM. Alternatively, just use HMAC on your ciphertexts. All of these things are relatively simple to deal with, though they certainly don't set the world on fire in terms of performance.

The one thing you should not do is say '*gosh this is complicated, I'll just use CBC mode and hope nobody attacks it',* at least not if you're building something that will potentially (someday) be online and subject to active attacks like the ones I described above. There's already enough stupid on the Internet, please don't add more.

Notes:

* Specifically, your encryption scheme must be IND-CPA secure, which would apply to CBC, CTR, CFB and OFB modes implemented with a secure block cipher. Your MAC must be existentially unforgeable under chosen message attack (EU-CMA), a property that's (believed) to be satisfied by most reasonable instantiations of HMAC.

** An earlier version of this post claimed that OCB didn't use Galois field arithmetic. This commenter on Reddit correctly points out that I'm an idiot. It does indeed do so. I stand by my point that the implementation is dramatically simpler than GCM.

16 comments:

Zooko May 19, 2012 at 11:56 PM

I believe the patents that Rogaway filed on OCB all expire in October of 2020.

There is also a patent by Jutla, perhaps now owned by IBM (I'm not sure) that might possibly apply to all of the authenticated encryption modes (excepting, of course, the generic composition approach that predated it). That one also expires in 2020 I guess.

Reply

Zooko May 19, 2012 at 11:57 PM

That latter one I mentioned is: 7,093,126 (Jutla, IBM, Aug 15, 2006), which I found out about from http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm#patent:phil

Reply

Replies

Matthew Green May 20, 2012 at 5:29 AM

That's scary. Hopefully nobody tries to enforce it.

Reply

Tom Ritter May 21, 2012 at 2:26 PM

There's also CCM*, a super-set of CCM, which is used by ZigBee in their standard. I believe CCM is coming TLS sometime soon, the draft was in Last Call a while ago. There was a IPR filing on it too: https://datatracker.ietf.org/ipr/1443/

Reply

CodeInChaos May 27, 2012 at 11:15 AM

Easy to use is what has a simple API and is part of the standard library. And that's exactly what we need for broad adoption. We need a fully specified(complete binary format including IV/nonce, padding, MAC) mode that's integrated in most popular platforms. And please don't expose the IV in the basic API.

Reply

JPGoldberg June 1, 2012 at 2:36 PM

If I'm stuck rolling my own Encrypt-then-MAC then

(1) what considerations should play into the length of the authentication tag
(2) are there key expansion mechanisms I can use that will allow me to save some space instead of having full independent keys for the MAC and the encryption?

Mostly, where can I read more about those sorts of considerations.

I've got space considerations as there will be many records in a database that will be encrypted.

Cheers,

-j

Reply

Unknown July 1, 2012 at 12:40 AM

David Wagner has a nice compare/contrst of AE(AD) modes at Slide 7 of http://www.cs.berkeley.edu/~daw/talks/FSE04eax.ppt. The slide includes criteria such as nonce sizes, single-key keying, static AD headers, and paralelization

Reply

cervone July 30, 2012 at 5:23 PM

Interesting article!

What about Synthetic Initialization Vector (SIV) AEAD (RFC5297) though?

Although rather complex, I like it because it explicitly builds from the assumption that nonce generation is easy only in theory; in practice it is hard or untrustable, and when it goes wrong it easily goes unnoticed.

For instance, if your software operates in a VM, very little entropy may be available to guarantee a random nonce.

On the other hand, several modes (e.g. GCM, which is CTR based) fails miserably if the nonce does fulfill the security requirements.

Reply

cervone July 30, 2012 at 5:24 PM

... if the nonce does NOT fulfill the security requirements.

Reply

Jack Winston December 4, 2012 at 6:02 PM

Great choice of topic! Deleting cookies from the server's memory is not enough nowadays, if this attack that they call submits chosen ciphertexts to some "decryption oracle" and analyzes the decrypted results tries to get an advantage against some information about the secret decryption key. Is this just a different version of "hacking"?

Reply

Anonymous January 10, 2013 at 9:51 AM

What about knwon weaknesses on GCM (see for instance http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf)?

Reply

Anonymous January 11, 2013 at 12:55 AM

OCB mode is now free for most software implementations. The license page linked has been updated.

Reply

S January 24, 2013 at 11:20 PM

OCB is now free for non-military uses! Check up at http://www.cs.ucdavis.edu/~rogaway/ocb/license.htm

Now I hope that Microsoft implements this into their crypto libraries ...

Reply

astel January 28, 2013 at 12:40 AM

How do you become this good? It's incredible to find out someone place such a lot of interest towards a topic. I am thankful I came across this.

Reply

Xiaofei Guo March 27, 2013 at 11:24 AM

Thank you for sharing this article. Why XCBC and IAPM are not secure?

Reply

william halimi June 15, 2013 at 7:37 AM

Very useful article & synthesis on AE subject. Thanks.

I have two comments & questions for Clarification:

- GCM : it appears as mentioned by Cervone to be highly sensitive (more than other AE modes ?) to IV uniqueness requirement and completely fails if such requirement is not respected'. So such weakness should'not be mentioned in the criteria for AE mode selection ? Remain GCM the best choice despite this weakness ?

- CCM : I understood via such mode review that it is based on MacThenEncrypt procedure (CBC-MAC then CTR) . so why such mode is retained as candidate if only Encrypt-Then-Mac procedure seems recommended in present article ?

Reply

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)