Ruby - Bug #10257

Generate X.509 certificate/request/CRL with elliptic curve keys

09/18/2014 03:24 PM - jtdowney (John Downey)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.0dev (2014-09-18 trunk 47624) [x86_64-darwin13]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

Elliptic curve keys (OpenSSL::PKey::EC) cannot currently be used with the X.509 classes in Ruby OpenSSL. This is due to a few slight incompatibilities between the way RSA/DSA are implemented and the way EC is implemented.

- OpenSSL::PKey::EC does not respond to #private? which is used by the #sign method on OpenSSL::X509::Certificate, OpenSSL::X509::Request, and OpenSSL::X509::CRL
- The #public_key method on OpenSSL::PKey::EC returns a OpenSSL::PKey::EC::Point instead of a OpenSSL::PKey::EC object with just public key fields

This patch adds an alias for #public? and #private? to OpenSSL::PKey::EC that correspond to #public_key? and #private_key?. This brings it in line with the same interface on OpenSSL::PKey::RSA and OpenSSL::PKey::DSA. This also allows the key to be used with the X.509 classes I mentioned.

The second issue is unfortunately more complex as it does not look like it is possible to fix without either breaking backwards compatibility or putting some branching deeper in OpenSSL::X509::Certificate, OpenSSL::X509::Request, and OpenSSL::X509::CRL. The good news is you can pass the private OpenSSL::PKey::EC key to #public_key= and it still does the right thing.

Related issues:

Related to Ruby - Bug #6567: Let OpenSSL::PKey::EC follow the general PKey in...	**Closed**

History

#1 - 10/03/2014 12:02 AM - bnagy (Ben Nagy)

I just ran across this issue as well. I monkey-patched in the private? method and was able to create the x509 cert, but it still doesn't appear to work for an OpenSSL connection (when assigned to a server context). Can you replicate this?

Here's a gist https://gist.github.com/bnagy/7a81e5387beeeea866c1 which works fine with an RSA key and fails with an EC key. I tried with an externally verified cert, which I have tested using the openssl s_server/s_client tools, as well as with an EC key that I pass to the ruby issue_cert method. I see:

SSL_accept returned=1 errno=0 state=SSLv3 read client hello C: no shared cipher
/Users/ben/.rubies/ruby-2.1.0/lib/ruby/2.1.0/openssl/ssl.rb:194:in `accept'

MRI: ruby 2.1.0p0 (2013-12-25 revision 44422) [x86_64-darwin12.0]

and

SSL_accept returned=1 errno=0 state=SSLv3 read client hello C: no shared cipher
/Users/ben/.rubies/rubinius-2.2.1/runtime/gems/rubysl-openssl-2.0.4/lib/openssl/ssl.rb:184:in `accept'

rubinius 2.2.1 (2.1.0 3ed43137 2013-11-17 JI) [x86_64-darwin12.4.0]

Can't test with JRuby because it doesn't support the ECDH suites at all yet.

Unfortunately, I haven't got any further yet because that's where the call vanishes into openssl itself, but I suspect 'no shared cipher' is a red herring (I'm not specifying or restricting any cipher suites at either end)

#2 - 12/17/2014 06:05 PM - ereslibre (Rafael Fernández López)

Just for reference. I also stepped on this issue, and there is an older bug about this issue: #5600

#3 - 09/13/2015 03:29 AM - zzak (zzak _)

- Assignee set to 7150

#4 - 03/25/2016 01:02 AM - tknarr (Todd Knarr)

As a developer trying to use OpenSSL::PKey::EC, I don't see a problem with breaking the #private_key and #public_key interfaces by changing the type of the return value. The values they return currently aren't usable as-is for anything but reimplementing the functionality OpenSSL::PKey::EC's supposed to provide, any code depending on them's probably broken already. My research also turned up almost no instances of anyone using these functions directly, everyone appears to use the openssl command itself to generate certificates and leaves verifying certificates and generating keys for connections up to the SSL library itself, which likely explains why there's so few complaints about this bug. I used the workaround described above and keys and certificates pass all the tests I can run after that.

The 'no shared cipher' bug is a separate issue involving the EC certificate not having the named-curve extension set. You can see it in the openssl x509 output for the certificate, without the extension the "Field Type" will be just "prime-field" (which allows for any curve, but OpenSSL itself can't handle arbitrary curves for SSL connections hence the error) while with the extension it'll be the name of the curve from #builtin_curves used to generate the original key. I'm working on what needs done to get that extension set using the existing Ruby API.

#5 - 04/09/2016 06:04 PM - tknarr (Todd Knarr)

"no shared cipher" for EC: looks like the curve name's in OpenSSL::PKey::EC::Group. There's a #curve_name method to get the curve name, but no way to set it and when a group's created using a named curve the name's never set in the resulting Group object. I haven't dug down far enough to confirm but it looks like the problem's in OpenSSL rather than the Ruby bindings.

#6 - 07/02/2016 02:22 AM - rhenium (Kazuki Yamaguchi)

- Related to Bug #6567: Let OpenSSL::PKey::EC follow the general PKey interface added

#7 - 07/02/2016 02:23 AM - rhenium (Kazuki Yamaguchi)

- Related to Bug #10497: OpenSSL Servers Do Not Support EC Certificates added

#8 - 07/02/2016 02:42 AM - rhenium (Kazuki Yamaguchi)

- Status changed from Open to Closed

This is fixed by r55098. (See also [Bug #6567]).

The 'no shared cipher' is another issue. ext/openssl didn't support ephemeral ECDH in server mode at that time.

#9 - 07/02/2016 07:38 AM - rhenium (Kazuki Yamaguchi)

- Related to deleted (Bug #10497: OpenSSL Servers Do Not Support EC Certificates)

Files

ec_x509.patch	8.06 KB	09/18/2014	jtdowney (John Downey)