Scatterlist Cryptographic API

Introduction

The Scatterlist Crypto API takes page vectors (scatterlists) as arguments, and works directly on pages. In some cases (e.g. ECB mode ciphers), this will allow for pages to be encrypted in-place with no copying.

One of the initial goals of this design was to readily support IPsec, so that processing can be applied to paged skb's without the need for linearization.

Details

At the lowest level are algorithms, which register dynamically with the API.

'Transforms' are user-instantiated objects, which maintain state, handle all of the implementation logic (e.g. manipulating page vectors) and provide an abstraction to the underlying algorithms. However, at the user level they are very simple.

Conceptually, the API layering looks like this:

```
[transform api]  (user interface)
[transform ops]  (per-type logic glue e.g. cipher.c, compress.c)
[algorithm api]  (for registering algorithms)
```

The idea is to make the user interface and algorithm registration API very simple, while hiding the core logic from both. Many good ideas from existing APIs such as Cryptoapi and Nettle have been adapted for this.

The API currently supports five main types of transforms: AEAD (Authenticated Encryption with Associated Data), Block Ciphers, Ciphers, Compressors and Hashes.

Please note that Block Ciphers is somewhat of a misnomer. It is in fact meant to support all ciphers including stream ciphers. The difference between Block Ciphers and Ciphers is that the latter operates on exactly one block while the former can operate on an arbitrary amount of data, subject to block size requirements (i.e., non-stream ciphers can only process multiples of blocks).

Here's an example of how to use the API:

```
#include <crypto/hash.h>
#include <linux/err.h>
#include <linux/scatterlist.h>

struct scatterlist sg[2];
char result[128];
struct crypto_ahash *tfm;
struct ahash_request *req;

tfm = crypto_alloc_ahash("md5", 0, CRYPTO_ALG_ASYNC);
if (IS_ERR(tfm))
        fail();

/* ... set up the scatterlists ... */

req = ahash_request_alloc(tfm, GFP_ATOMIC);
if (!req)
        fail();

ahash_request_set_callback(req, 0, NULL, NULL);
ahash_request_set_crypt(req, sg, result, 2);

if (crypto_ahash_digest(req))
        fail();

ahash_request_free(req);
crypto_free_ahash(tfm);
```

Many real examples are available in the regression test module (tcrypt.c).

Developer Notes

Transforms may only be allocated in user context, and cryptographic methods may only be called from softirq and user contexts. For transforms with a setkey method it too should only be called from user context.

When using the API for ciphers, performance will be optimal if each scatterlist contains data which is a multiple of the cipher's block size (typically 8 bytes). This prevents having to do any copying across non-aligned page fragment boundaries.

Adding New Algorithms

When submitting a new algorithm for inclusion, a mandatory requirement is that at least a few test vectors from known sources (preferably standards) be included.

Converting existing well known code is preferred, as it is more likely to have been reviewed and widely tested. If submitting code from LGPL sources, please consider changing the license to GPL (see section 3 of the LGPL).

Algorithms submitted must also be generally patent-free (e.g. IDEA will not be included in the mainline until around 2011), and be based on a recognized standard and/or have been subjected to appropriate peer review.

Also check for any RFCs which may relate to the use of specific algorithms, as well as general application notes such as RFC2451 ("The ESP CBC-Mode Cipher Algorithms").

It's a good idea to avoid using lots of macros and use inlined functions instead, as gcc does a good job with inlining, while excessive use of macros can cause compilation problems on some platforms.

Also check the TODO list at the web site listed below to see what people might already be working on.

Bugs

Send bug reports to:
linux-crypto@vger.kernel.org
Cc:
Herbert Xu <herbert@gondor.apana.org.au>, David S. Miller <davem@redhat.com>

Further Information

For further patches and various updates, including the current TODO list, see: http://gondor.apana.org.au/~herbert/crypto/

Authors

- James Morris
- David S. Miller
- Herbert Xu

Credits

The following people provided invaluable feedback during the development of the API:

- Alexey Kuznetzov
- Rusty Russell
- Herbert Valerio Riedel
- Jeff Garzik
- Michael Richardson
- Andrew Morton
- Ingo Oeser
- Christoph Hellwig

Portions of this API were derived from the following projects:

Kerneli Cryptoapi (http://www.kerneli.org/)
- Alexander Kjeldaas
- Herbert Valerio Riedel
- Kyle McMartin
- Jean-Luc Cooke
- David Bryson
- Clemens Fruhwirth
- Tobias Ringstrom
- Harald Welte

and;

Nettle (https://www.lysator.liu.se/~nisse/nettle/)
- Niels MÃ¶ller

Original developers of the crypto algorithms:

- Dana L. How (DES)
- Andrew Tridgell and Steve French (MD4)
- Colin Plumb (MD5)

- Steve Reid (SHA1)
- Jean-Luc Cooke (SHA256, SHA384, SHA512)
- Kazunori Miyazawa / USAGI (HMAC)
- Matthew Skala (Twofish)
- Dag Arne Osvik (Serpent)
- Brian Gladman (AES)
- Kartikey Mahendra Bhatt (CAST6)
- Jon Oberheide (ARC4)
- Jouni Malinen (Michael MIC)
- NTT(Nippon Telegraph and Telephone Corporation) (Camellia)

SHA1 algorithm contributors:

- Jean-Francois Dive

DES algorithm contributors:

- Raimar Falke
- Gisle SÃ¦lensminde
- Niels MÃ¶ller

Blowfish algorithm contributors:

- Herbert Valerio Riedel
- Kyle McMartin

Twofish algorithm contributors:

- Werner Koch
- Marc Mutz

SHA256/384/512 algorithm contributors:

- Andrew McDonald
- Kyle McMartin
- Herbert Valerio Riedel

AES algorithm contributors:

- Alexander Kjeldaas
- Herbert Valerio Riedel
- Kyle McMartin
- Adam J. Richter
- Fruhwirth Clemens (i586)
- Linus Torvalds (i586)

CAST5 algorithm contributors:

- Kartikey Mahendra Bhatt (original developers unknown, FSF copyright).

TEA/XTEA algorithm contributors:

- Aaron Grothe
- Michael Ringe

Khazad algorithm contributors:

- Aaron Grothe

Whirlpool algorithm contributors:

- Aaron Grothe
- Jean-Luc Cooke

Anubis algorithm contributors:

- Aaron Grothe

Tiger algorithm contributors:

- Aaron Grothe

VIA PadLock contributors:

- Michal Ludvig

Camellia algorithm contributors:

- NTT(Nippon Telegraph and Telephone Corporation) (Camellia)

Generic scatterwalk code by Adam J. Richter <adam@yggdrasil.com>

Please send any credits updates or corrections to: Herbert Xu <herbert@gondor.apana.org.au>