Windows 版 GnuPG 2.1.x を使ってみる

GnuPG (https://www.gnupg.org/)2.1.x(modern version)では Windows 用のバイナリも公式サイトで提供されている。

- GnuPG - Download (https://www.gnupg.org/download/)
- GnuPG - Manuals (https://www.gnupg.org/documentation/manuals.html)

Modern version は RFC6637 (http://tools.ietf.org/html/rfc6637)で定義される ECC(Elliptic Curve Cryptography, 楕円曲線暗号)をサポートしているので，さっそく試してみる。ちなみに ECC の原理については以下のページが参考になる。

- 暗号の話 - 楕円曲線暗号 (http://mailsrv.nara-edu.ac.jp/~asait/crypt.htm#section44)

今回は Windows 版を使ったけど，端末でコマンドを叩く分にはどの OS でも使い方は同じだと思うので，よかったら参考にどうぞ。

準備

今回使った GnuPG (https://www.gnupg.org/)のバージョンは以下の通り。

```
C:>gpg --version
gpg (GnuPG) 2.1.1
libgcrypt 1.6.2
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ********
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

あと，鍵を生成するにあたってキャラクタ設定をしておく。

```
Real name: John Doe
Email address: john@examle.com
```

ECC 鍵の生成

通常 GnuPG (https://www.gnupg.org/)で鍵を生成する際には `--gen-key` オプションを使うのだが，modern version では `--gen-key` オプションを使った鍵生成手順が大幅に簡略化されていて，公開鍵暗号のアルゴリズムも RSA/2014bits に固定されている。

```
C:>gpg --gen-key
gpg (GnuPG) 2.1.1; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-gen-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: John Doe
Email address: john@examle.com
You selected this USER-ID:
    "John Doe <john@examle.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key CC0EBA52 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   4  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 4u
gpg: depth: 1  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 1f, 0u
gpg: next trustdb check due at 2015-11-19
pub   rsa2048/CC0EBA52 2015-01-23
      Key fingerprint = CAC7 1FE9 5B3F 5015 B8C6  A2D4 42D8 C73E CC0E BA52
uid       [ultimate] John Doe <john@examle.com>
sub   rsa2048/D479529E 2015-01-23
```

```
C:>
```

他のタイプの鍵を生成する場合，あるいは生成時に詳細な設定を行いたい場合は `--full-gen-key` オプションを使う。

```
C:>gpg --full-gen-key
gpg (GnuPG) 2.1.1; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection?
```

ECC 鍵を生成したい場合は `--expert` オプションも組み合わせる。

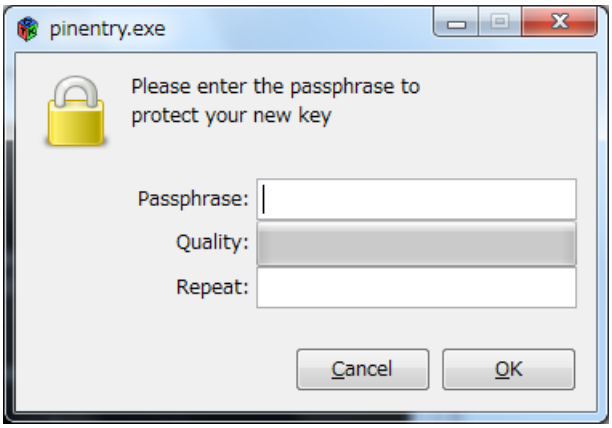
```
C:>gpg --full-gen-key --expert
gpg (GnuPG) 2.1.1; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
   (9) ECC and ECC
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
Your selection?
```

今回は ECC を生成したいので (9) を選択する。ではスタート！

```
C:>gpg --full-gen-key --expert
gpg (GnuPG) 2.1.1; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
   (9) ECC and ECC
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
Your selection? 9
Please select which elliptic curve you want:
   (2) NIST P-256
   (3) NIST P-384
   (4) NIST P-521
   (5) Brainpool P-256
   (6) Brainpool P-384
   (7) Brainpool P-512
Your selection? 2
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 1w
Key expires at 01/31/15 11:20:43 東京 (標準時)
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: John Doe
Email address: john@examle.com
Comment: forECC
You selected this USER-ID:
    "John Doe (forECC) <john@examle.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
```

```
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 5FBBFA18 marked as ultimately trusted
gpg: directory 'C:/pathto/openpgp-revocs.d' created
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: Note: signatures using the MD5 algorithm are rejected
gpg: depth: 0  valid:  13  signed:  10  trust: 0-, 0q, 0n, 0m, 0f, 13u
gpg: depth: 1  valid:   8  signed:   4  trust: 4-, 0q, 0n, 3m, 1f, 0u
gpg: next trustdb check due at 2015-01-31
pub   nistp256/5FBBFA18 2015-01-24 [expires: 2015-01-31]
      Key fingerprint = 8B20 19D4 33A7 729E 5BFE  5779 31FB FDA9 5FBB FA18
uid       [ultimate] John Doe (forECC) <john@examle.com>
sub   nistp256/252C4D79 2015-01-24 nistp256 [expires: 2015-01-31]


C:>
```

途中でパスフレーズを訊かれる（Windows 版であれば以下のプロンプトがポップアップする）ので，適切なパスフレーズをセットすること。（パスフレーズについては拙文「わかる！ OpenPGP 暗号 (http://www.baldanders.info/spiegel/archive/pgpdump/openpgp.shtml#passphrase)」を参考にどうぞ）



お試しなので有効期間を1週間とした。

ECC 鍵の生成方法として以下の6つのオプションが提示される。

```
   (2) NIST P-256
   (3) NIST P-384
   (4) NIST P-521
   (5) Brainpool P-256
   (6) Brainpool P-384
   (7) Brainpool P-512
```

NIST P-xxx は FIPS PUB 186-4 (http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf)で示される方式で鍵を生成している。Brainpool P-xxx は Brainpool (http://www.ecc-brainpool.org/)と呼ばれる方式で鍵を生成している（Brainpool (http://www.ecc-brainpool.org/)は RFC5639 (http://tools.ietf.org/html/rfc5639)で標準化されている）。各数字部分は鍵長を表す。

ここでは NIST P-256 を選択した。（鍵長の選択については拙文「わかる！ OpenPGP 暗号 (http://www.baldanders.info/spiegel/archive/pgpdump/openpgp.shtml#key-size)」を参考にどうぞ）

さて，生成した ECC の公開鍵がこちら。

```
C:>gpg --armor --export john@examle.com
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mFIEVMMBvxMIKoZIzj0DAQcCAwSl1bx2B9u4j7IhGRGw0Fp+6TTfo42O+bl+tthj
Cu6S7g10x8BI87jVqqhzvecZtdrY9mgFAxV9moRDYcru39YOtCNKb2huIERvZSAo
Zm9yRUNDKSA8am9obkBleGFtbGUuY29tPoh/BBMICAAnBQJUwwG/AhsDBQkACTqA
BQsJCAcCBhUICQoLAgQWAgMBAh4BAheAAAoJEDH7/alfu/oYSaQA/AlgoAnPVyVH
Y38j/ebL7+bbe0ulfy8sj4W4TgWctj3TAQDlRAuT+bcTISTvjghtTkg/BP55LU19
xsHC1lc4M77zErhWBFTDAb8SCCqGSM49AwEHAgMEjD3FoH0SRJmBgMDbmgquHXQH
xoU6Yr3WVIfWLLkr+4l9bSDjmVnSCXMVR/NOpEmn1aN5alt1x/h7XPwofz7L0QMB
CAeIZwQYEwgADwUCVMMBvwIbDAUJAAk6gAAKCRAx+/2pX7v6GMYnAQDAL3ZaEG0d
Ih5iwZu8YtEGSvE6R1rpC/E5bONnoJY80gEAjFkcOoUOHtOYRRNNMOK5pBUOG21m
Gqfn1eJRB5Vgh5E=
=RME9
-----END PGP PUBLIC KEY BLOCK-----
```

これを pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)で見てみると

```
C:>gpg --armor --export john@examle.com | pgpdump
```

```
Old: Public Key Packet(tag 6)(82 bytes)
        Ver 4 - new
        Public key creation time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Pub alg - Reserved for ECDSA(pub 19)
        Unknown public key(pub 19)
Old: User ID Packet(tag 13)(35 bytes)
        User ID - John Doe (forECC) <john@examle.com>
Old: Signature Packet(tag 2)(127 bytes)
        Ver 4 - new
        Sig type - Positive certification of a User ID and Public Key packet(0x13).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Hashed Sub: key flags(sub 27)(1 bytes)
                Flag - This key may be used to certify other keys
                Flag - This key may be used to sign data
        Hashed Sub: key expiration time(sub 9)(4 bytes)
                Time - Sat Jan 31 11:21:51 東京 (標準時) 2015
        Hashed Sub: preferred symmetric algorithms(sub 11)(4 bytes)
                Sym alg - AES with 256-bit key(sym 9)
                Sym alg - AES with 192-bit key(sym 8)
                Sym alg - AES with 128-bit key(sym 7)
                Sym alg - Triple-DES(sym 2)
        Hashed Sub: preferred hash algorithms(sub 21)(5 bytes)
                Hash alg - SHA256(hash 8)
                Hash alg - SHA384(hash 9)
                Hash alg - SHA512(hash 10)
                Hash alg - SHA224(hash 11)
                Hash alg - SHA1(hash 2)
        Hashed Sub: preferred compression algorithms(sub 22)(3 bytes)
                Comp alg - ZLIB <RFC1950>(comp 2)
                Comp alg - BZip2(comp 3)
                Comp alg - ZIP <RFC1951>(comp 1)
        Hashed Sub: features(sub 30)(1 bytes)
                Flag - Modification detection (packets 18 and 19)
        Hashed Sub: key server preferences(sub 23)(1 bytes)
                Flag - No-modify
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - 49 a4
        Unknown signature(pub 19)
Old: Public Subkey Packet(tag 14)(86 bytes)
        Ver 4 - new
        Public key creation time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Pub alg - Reserved for Elliptic Curve(pub 18)
        Unknown public key(pub 18)
Old: Signature Packet(tag 2)(103 bytes)
        Ver 4 - new
        Sig type - Subkey Binding Signature(0x18).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Hashed Sub: key flags(sub 27)(1 bytes)
                Flag - This key may be used to encrypt communications
                Flag - This key may be used to encrypt storage
        Hashed Sub: key expiration time(sub 9)(4 bytes)
                Time - Sat Jan 31 11:21:51 東京 (標準時) 2015
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - c6 27
        Unknown signature(pub 19)
```

となる。主鍵(署名鍵)が ECDSA(pub19), 副鍵(暗号鍵)が ECDH (pub18) になっているのがお分かりだろうか。

(pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)はまだ RFC6637 (http://tools.ietf.org/html/rfc6637)に対応していないので上手く dump できてないがご容赦。ちなみに Windows 用のバイナリは私のサイトに置いてある (http://www.baldanders.info/spiegel/archive/pgpdump/)ので，是非どうぞ)

副鍵を含めた電子指紋を見たいときは `--fingerprint` を重ねて指定するとよい。

```
C:>gpg --fingerprint --fingerprint john@examle.com
pub   nistp256/5FBBFA18 2015-01-24 [expires: 2015-01-31]
      Key fingerprint = 8B20 19D4 33A7 729E 5BFE  5779 31FB FDA9 5FBB FA18
uid       [ultimate] John Doe (forECC) <john@examle.com>
sub   nistp256/252C4D79 2015-01-24 nistp256 [expires: 2015-01-31]
      Key fingerprint = 8F2F D10A 4D10 BF69 1D1D  8086 EE06 6BFE 252C 4D79
```

データをやりとりする相手に渡す公開鍵は `--export` で出力したものを渡す。また公開鍵サーバに鍵を送る場合は `--send-keys` オプションを使う。(`--send-keys` オプションを使う際は john@examle.com などのユーザIDではなく 0x5FBBFA18 などの鍵IDを指定する)

```
C:>gpg --send-keys 0x5FBBFA18
gpg: sending key 5FBBFA18 to hkp server keys.gnupg.net
```

```
C:>
```

鍵サーバから公開鍵を取得する際は --search-keys または --recv-keys オプションを使う。(--recv-keys オプションを使う際も鍵IDを指定する。あらかじめ鍵IDがわからない場合は --search-keys でユーザIDを指定して検索できる)

```
C:>gpg --search-keys john@examle.com
gpg: data source: http://79.143.214.216:11371
(1)     John Doe (forECC) <john@examle.com>
          256 bit ECDSA key 5FBBFA18, created: 2015-01-24
Keys 1-1 of 1 for "john@examle.com".  Enter number(s), N)ext, or Q)uit > q

C:>gpg --recv-keys 0x5FBBFA18
gpg: key 5FBBFA18: "John Doe (forECC) <john@examle.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1

C:>
```

電子署名

最初に電子署名の対象となるテキストファイルを用意する。

```
C:>echo Hello World! > hello.txt
```

以降, このファイルに対して署名を行う。

クリア署名

まずはクリア署名から。

```
C:>gpg --clearsign --local-user john@examle.com hello.txt
```

hello.txt の署名結果が hello.txt.asc に出力される。hello.txt.asc の内容は以下のようになる。

```
C:>type hello.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Hello World!
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iF4EARMIAAYFAlTDCN8ACgkQMfv9qV+7+hg2HwEA6h2iFFuCBv3VrsSf2BREQaT1
T1ZprZqwRPOjiLJg9AwA/ArTwCPz7c2vmxlv7sRlRLUI6CdsOqhuO1KfYXrq7idI
=ZOTN
-----END PGP SIGNATURE-----
```

(type はコマンドプロンプトの組込みコマンドで UNIX 系の cat コマンドと同等)

このようにクリア署名では元のテキストと電子署名が一体になっている。署名者は hello.txt.asc の内容を相手に送ればよい。

この署名の pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)をとってみる。

```
C:>pgpdump hello.txt.asc
Old: Signature Packet(tag 2)(94 bytes)
        Ver 4 - new
        Sig type - Signature of a canonical text document(0x01).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:52:15 東京 (標準時) 2015
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - 36 1f
        Unknown signature(pub 19)
```

このように電子署名には時刻情報も含まれていることがお分かりだろう。

この署名を検証してみる。署名の検証には --verify を使う。

```
C:>gpg --verify hello.txt.asc
gpg: Signature made 01/24/15 11:52:15 東京 (標準時) using ECDSA key ID 5FBBFA18
gpg: Good signature from "John Doe (forECC) <john@examle.com>" [ultimate]
gpg: WARNING: not a detached signature; file 'hello.txt' was NOT verified!
```

クリア署名の場合は最後の WARNING は気にしなくてもよい。検証は --decrypt オプションでも可能だが結果がちょっと異なる。

```
C:>gpg --decrypt hello.txt.asc
Hello World!
gpg: Signature made 01/24/15 11:52:15 東京 (標準時) using ECDSA key ID 5FBBFA18
gpg: Good signature from "John Doe (forECC) <john@examle.com>" [ultimate]
```

署名対象のテキストが表示されるのでこちらのほうが便利かな。

分離署名

元のファイルと署名を分けたいときは分離署名にする。

```
C:>gpg --detach-sign --local-user john@examle.com hello.txt
```

署名は hello.txt.sig に出力される。

```
C:>gpg --armor --detach-sign --local-user john@examle.com hello.txt
```

--armor オプションを付けた場合は，署名が hello.txt.asc に出力される。hello.txt.sig と hello.txt.asc の違いは，前者がバイナリファイルで後者がテキストファイルである。

hello.txt.asc の中身はこんな感じ。

```
C:>type hello.txt.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iF4EABMIAAYFAlTDHMMACgkQMfv9qV+7+hhp5wD/b1TzfX0hPO2QH9bWjRFPnG4s
IAU+nzN9WSNurIskSYYBAMJ8WxSA/3udJe2g+h8fnl/XGk7qapqx6xTmCjhaUP4S
=LJYe
-----END PGP SIGNATURE-----
```

署名部分だけが出力されているのがお分かりだろうか。この署名の pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)をとってみると

```
C:>pgpdump hello.txt.asc
Old: Signature Packet(tag 2)(94 bytes)
        Ver 4 - new
        Sig type - Signature of a binary document(0x00).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 13:17:07 東京 (標準時) 2015
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - 69 e7
        Unknown signature(pub 19)
```

となる。

電子署名を検証するには電子署名対象の hello.txt と hello.txt.asc(または hello.txt.sig)を同じフォルダにおいて

```
C:>gpg --verify hello.txt.asc
gpg: assuming signed data in 'hello.txt'
gpg: Signature made 01/24/15 13:17:07 東京 (標準時) using ECDSA key ID 5FBBFA18
gpg: Good signature from "John Doe (forECC) <john@examle.com>" [ultimate]
```

とする(署名ファイルのファイル名から署名対象のファイル名を推測してくれる)。あるいは署名対象を明示して

```
C:>gpg --verify hello.txt.asc hello.txt
gpg: Signature made 01/24/15 13:17:07 東京 (標準時) using ECDSA key ID 5FBBFA18
gpg: Good signature from "John Doe (forECC) <john@examle.com>" [ultimate]
```

でもよい。

テキストモードとバイナリモード

GnuPG (https://www.gnupg.org/)の電子署名には「テキストモード」と「バイナリモード」がある。クリア署名の際は自動的にテキストモードで署名し，分離署名の際は自動的にバイナリモードで署名する。両者の違いは，バイナリモードでは文字通り署名対象データをバイナリデータとして扱うのに対し，テキストモードでは署名対象をテキストとみなし内部で改行コードを CRLF に変換する。テキストデータは，転送する際に改行が OS ごとのコードに書き換えられることが多いため，このような仕組みが必要になる。

分離署名の際，強制的にテキストモードで署名を行いたい場合は --textmode オプションを付加する(メール暗号化の PGP/MIME で必要なオプション)。一方，--textmode を打ち消すオプションとして --no-textmode も用意されている。

データの暗号化と復号

引き続き hello.txt を使って，今度は暗号化と復号を行ってみる。

暗号化

hello.txt を暗号化するには --encrypt オプションを使う。

```
C:>gpg --encrypt --recipient john@examle.com hello.txt
```

暗号化データは hello.txt.gpg に出力される。

```
C:>gpg --armor --encrypt --recipient john@examle.com hello.txt
```

--armor オプションを付けた場合は，暗号化データが hello.txt.asc に出力される。hello.txt.gpg と hello.txt.asc の違いは，前者がバイナリファイルで後者がテキストファイルである。

(分かりにくいかもしれないので補足しておくと，暗号化の際には暗号化データの受取人を --recipient オプションで指定する。暗号化データを復号できるのは --recipient で指定したユーザだけである。受取人が複数ある場合は必要なだけ受取人を指定しなければならない。また，自分自身も復号可能にするには受取人に自分も含める必要がある)

hello.txt.asc の中身はこんな感じ。

```
C:>type hello.txt.asc
```

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2

hH4D7gZr/iUsTXkSAgMEw+fXK68lKhn2J4DqfE9tymEiWuOtDPvZotWkMJrz7jRU
rqj2RqyKrjimT/Py7jBAYlsH5yvumpDUbx7XwyYhqzBK/oiinw6r8756iScyOLgG
dfysPNS6D0lkFapImtvBinsRdvsv7++wKakkdW1pEk3SUQFq5nHK//ax/z9xyHdF
iFH/4/LDlVfnKYDo5YZ86pj0BLOK+IjIkfdWe8utdUBI0Vo/Pywd5Da76fd3srgq
RAO+eOIFO0S22E5hpUMFdoo8ZA==
=5cN+
-----END PGP MESSAGE-----
```

この暗号データの pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)をとってみると

```
C:>pgpdump hello.txt.asc
Old: Public-Key Encrypted Session Key Packet(tag 1)(126 bytes)
        New version(3)
        Key ID - 0xEE066BFE252C4D79
        Pub alg - Reserved for Elliptic Curve(pub 18)
                unknown(pub 18)
                -> m = sym alg(1 byte) + checksum(2 bytes) + PKCS-1 block type 02
New: Symmetrically Encrypted and MDC Packet(tag 18)(81 bytes)
        Ver 1
        Encrypted data [sym alg is specified in pub-key encrypted session key]
                (plain text + MDC SHA1(20 bytes))
```

となる。

暗号化データの復号

復号には --decrypt オプションを使う。

```
C:>gpg --output hello.txt.out --decrypt hello.txt.asc
gpg: encrypted with 256-bit ECDH key, ID 252C4D79, created 2015-01-24
      "John Doe (forECC) <john@examle.com>"
```

注意

GnuPG (https://www.gnupg.org/)では復号結果を標準出力に出力する。Linux などの UNIX 系の shell では復号結果をファイルにリダイレクトするなり次の処理にパイプするなりできるが，Windows ではデータを標準出力に出力すると改行コードを自動変換（LF→CRLF など）してしまうアレな仕様になっているためデータの内容が変わってしまう。このため，Windows で暗号化データを復号する際には必ず --output オプションで出力先ファイルを指定すること。

暗号化＋電子署名

GnuPG (https://www.gnupg.org/)では暗号化と電子署名を同時に行うことができる。

```
C:>gpg --armor --sign --encrypt --recipient alice@examle.com --local-user john@examle.com hello.txt
```

これを受取人の alice@examle.com が復号すると

```
C:>gpg --output hello.txt.out --decrypt hello.txt.asc
gpg: encrypted with 3072-bit ELG key, ID 54D7887B, created 2015-01-24
      "Alice (DEMO) <alice@examle.com>"
gpg: Signature made 01/24/15 15:15:40 東京 (標準時) using ECDSA key ID 5FBBFA18
gpg: Good signature from "John Doe (forECC) <john@examle.com>" [  full  ]
```

となり，復号と署名の検証を同時に行う。

鍵の破棄

生成した鍵を破棄しなければならないことがある。例えば暗号アルゴリズムが危殆化して使えなくなったとか，秘密鍵が NSA に盗られてしまったとか，などである。

OpenPGP では，あらかじめ「失効証明書」を作成し，失効証明書を読み込むことで鍵を破棄できる。

GnuPG (https://www.gnupg.org/)modern version では，鍵の生成時に失効証明書も自動で生成してくれる。失効証明書は鍵束のあるフォルダの openpgp-revocs.d フォルダ内に作成されている。

失効証明書のファイル名は〈電子指紋〉.rev。たとえば john@examle.com の鍵の電子指紋が

```
C:>gpg --fingerprint john@examle.com
pub   nistp256/5FBBFA18 2015-01-24 [expires: 2015-01-31]
      Key fingerprint = 8B20 19D4 33A7 729E 5BFE  5779 31FB FDA9 5FBB FA18
uid       [ultimate] John Doe (forECC) <john@examle.com>
sub   nistp256/252C4D79 2015-01-24 nistp256 [expires: 2015-01-31]
```

なら openpgp-revocs.d/8B2019D433A7729E5BFE577931FBFDA95FBBFA18.rev が失効証明書である。

失効証明書の中身は以下のようになっている。

```
C:>type 8B2019D433A7729E5BFE577931FBFDA95FBBFA18.rev
This is a revocation certificate for the OpenPGP key:

pub  nistp256/5FBBFA18 2015-01-24
      Key fingerprint = 8B20 19D4 33A7 729E 5BFE  5779 31FB FDA9 5FBB FA18
uid                  John Doe (forECC) <john@examle.com>
```

```
Use it to revoke this key in case of a compromise or loss of
the secret key.  However, if the secret key is still accessible,
it is better to generate a new revocation certificate and give
a reason for the revocation.

To avoid an accidental use of this file, a colon has been inserted
before the 5 dashes below.  Remove this colon with a text editor
before making use of this revocation certificate.

:-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2
Comment: This is a revocation certificate

iGEEIBMIAAkFAlTDAcwCHQAACgkQMfv9qV+7+hjjeQD/eLVZ8NMqeMFxrLQ+LFVQ
wZnQZsIUhKvVQVw6RdC8j6QBAJSlJGF3YawIiYbPJ8MqqqB/7AbtiPQMX0YUsSqn
YimK
=lgOW
-----END PGP PUBLIC KEY BLOCK-----
```

:-----BEGIN PGP PUBLIC KEY BLOCK----- の先頭のコロン(:)を外すことで失効証明書が有効になる。

ちなみにこれを(コロンを外した上で) pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)**でみると**

```
C:>pgpdump 8B2019D433A7729E5BFE577931FBFDA95FBBFA18.rev
Old: Signature Packet(tag 2)(97 bytes)
        Ver 4 - new
        Sig type - Key revocation signature(0x20).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:22:04 東京 (標準時) 2015
        Hashed Sub: reason for revocation(sub 29)(1 bytes)
                Reason - No reason specified
                Comment -
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - e3 79
        Unknown signature(pub 19)
```

となっている。

失効証明書を読み込むには `--import` **オプションを使う。**

```
C:>gpg --import 8B2019D433A7729E5BFE577931FBFDA95FBBFA18.rev
gpg: key 5FBBFA18: "John Doe (forECC) <john@examle.com>" revocation certificate imported
gpg: Total number processed: 1
gpg:    new key revocations: 1
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: Note: signatures using the MD5 algorithm are rejected
gpg: depth: 0  valid:  14  signed:  10  trust: 0-, 0q, 0n, 0m, 0f, 14u
gpg: depth: 1  valid:   8  signed:   4  trust: 4-, 0q, 0n, 3m, 1f, 0u
gpg: next trustdb check due at 2015-01-31

C:>gpg --fingerprint john@examle.com
pub   nistp256/5FBBFA18 2015-01-24 [revoked: 2015-01-24]
      Key fingerprint = 8B20 19D4 33A7 729E 5BFE  5779 31FB FDA9 5FBB FA18
uid       [ revoked] John Doe (forECC) <john@examle.com>
```

pgpdump (http://www.mew.org/~kazu/proj/pgpdump/)**でみると**

```
C:>gpg --armor --export john@examle.com | pgpdump
Old: Public Key Packet(tag 6)(82 bytes)
        Ver 4 - new
        Public key creation time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Pub alg - Reserved for ECDSA(pub 19)
        Unknown public key(pub 19)
Old: Signature Packet(tag 2)(97 bytes)
        Ver 4 - new
        Sig type - Key revocation signature(0x20).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:22:04 東京 (標準時) 2015
        Hashed Sub: reason for revocation(sub 29)(1 bytes)
                Reason - No reason specified
                Comment -
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - e3 79
        Unknown signature(pub 19)
Old: User ID Packet(tag 13)(35 bytes)
        User ID - John Doe (forECC) <john@examle.com>
Old: Signature Packet(tag 2)(127 bytes)
```

```
        Ver 4 - new
        Sig type - Positive certification of a User ID and Public Key packet(0x13).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Hashed Sub: key flags(sub 27)(1 bytes)
                Flag - This key may be used to certify other keys
                Flag - This key may be used to sign data
        Hashed Sub: key expiration time(sub 9)(4 bytes)
                Time - Sat Jan 31 11:21:51 東京 (標準時) 2015
        Hashed Sub: preferred symmetric algorithms(sub 11)(4 bytes)
                Sym alg - AES with 256-bit key(sym 9)
                Sym alg - AES with 192-bit key(sym 8)
                Sym alg - AES with 128-bit key(sym 7)
                Sym alg - Triple-DES(sym 2)
        Hashed Sub: preferred hash algorithms(sub 21)(5 bytes)
                Hash alg - SHA256(hash 8)
                Hash alg - SHA384(hash 9)
                Hash alg - SHA512(hash 10)
                Hash alg - SHA224(hash 11)
                Hash alg - SHA1(hash 2)
        Hashed Sub: preferred compression algorithms(sub 22)(3 bytes)
                Comp alg - ZLIB <RFC1950>(comp 2)
                Comp alg - BZip2(comp 3)
                Comp alg - ZIP <RFC1951>(comp 1)
        Hashed Sub: features(sub 30)(1 bytes)
                Flag - Modification detection (packets 18 and 19)
        Hashed Sub: key server preferences(sub 23)(1 bytes)
                Flag - No-modify
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - 49 a4
        Unknown signature(pub 19)
Old: Public Subkey Packet(tag 14)(86 bytes)
        Ver 4 - new
        Public key creation time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Pub alg - Reserved for Elliptic Curve(pub 18)
        Unknown public key(pub 18)
Old: Signature Packet(tag 2)(103 bytes)
        Ver 4 - new
        Sig type - Subkey Binding Signature(0x18).
        Pub alg - Reserved for ECDSA(pub 19)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Sat Jan 24 11:21:51 東京 (標準時) 2015
        Hashed Sub: key flags(sub 27)(1 bytes)
                Flag - This key may be used to encrypt communications
                Flag - This key may be used to encrypt storage
        Hashed Sub: key expiration time(sub 9)(4 bytes)
                Time - Sat Jan 31 11:21:51 東京 (標準時) 2015
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x31FBFDA95FBBFA18
        Hash left 2 bytes - c6 27
        Unknown signature(pub 19)
```

となっている。署名パケット(tag 2)のタイプが “Key revocation signature(0x20).” なっているのがお分かりだろうか。

鍵サーバにある公開鍵を破棄するには，手元の鍵を破棄した上で，破棄した鍵を鍵サーバに送る。

```
C:>gpg --send-keys 0x5FBBFA18
gpg: sending key 5FBBFA18 to hkp server keys.gnupg.net

C:>gpg --search-keys john@examle.com
gpg: data source: http://79.143.214.216:11371
(1)     John Doe (forECC) <john@examle.com>
          256 bit ECDSA key 5FBBFA18, created: 2015-01-24 (revoked)
Keys 1-1 of 1 for "john@examle.com".  Enter number(s), N)ext, or Q)uit >
```

失効証明書はいざというときの最後の切り札なので，無くさないよう大事に保管しておいて欲しい。昔だったら「フロッピーにコピー」だったのだが，今はどうなんだろう。やはり CD-R か DVD に焼くのがいいだろうか。

GnuPG オプション一覧

GnuPG (https://www.gnupg.org/)のオプションや簡単な使い方は `--help` オプションで見ることができる。

```
C:>gpg --help
gpg (GnuPG) 2.1.1
libgcrypt 1.6.2
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
```

```
There is NO WARRANTY, to the extent permitted by law.

Home: ********
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2

Syntax: gpg [options] [files]
Sign, check, encrypt or decrypt
Default operation depends on the input data

Commands:

 -s, --sign                make a signature
     --clearsign           make a clear text signature
 -b, --detach-sign         make a detached signature
 -e, --encrypt             encrypt data
 -c, --symmetric           encryption only with symmetric cipher
 -d, --decrypt             decrypt data (default)
     --verify              verify a signature
 -k, --list-keys           list keys
     --list-sigs           list keys and signatures
     --check-sigs          list and check key signatures
     --fingerprint         list keys and fingerprints
 -K, --list-secret-keys    list secret keys
     --gen-key             generate a new key pair
     --quick-gen-key       quickly generate a new key pair
     --full-gen-key        full featured key pair generation
     --gen-revoke          generate a revocation certificate
     --delete-keys         remove keys from the public keyring
     --delete-secret-keys  remove keys from the secret keyring
     --quick-sign-key      quickly sign a key
     --quick-lsign-key     quickly sign a key locally
     --sign-key            sign a key
     --lsign-key           sign a key locally
     --edit-key            sign or edit a key
     --passwd              change a passphrase
     --export              export keys
     --send-keys           export keys to a key server
     --recv-keys           import keys from a key server
     --search-keys         search for keys on a key server
     --refresh-keys        update all keys from a keyserver
     --import              import/merge keys
     --card-status         print the card status
     --card-edit           change data on a card
     --change-pin          change a card's PIN
     --update-trustdb      update the trust database
     --print-md            print message digests
     --server              run in server mode

Options:

 -a, --armor               create ascii armored output
 -r, --recipient USER-ID   encrypt for USER-ID
 -u, --local-user USER-ID  use USER-ID to sign or decrypt
 -z N                      set compress level to N (0 disables)
     --textmode            use canonical text mode
 -o, --output FILE         write output to FILE
 -v, --verbose             verbose
 -n, --dry-run             do not make any changes
 -i, --interactive         prompt before overwriting
     --openpgp             use strict OpenPGP behavior

(See the man page for a complete listing of all commands and options)

Examples:

 -se -r Bob [file]         sign and encrypt for user Bob
 --clearsign [file]        make a clear text signature
 --detach-sign [file]      make a detached signature
 --list-keys [names]       show keys
 --fingerprint [names]     show fingerprints

Please report bugs to <http://bugs.gnupg.org>.

C:>
```

ここで紹介したオプションについていくつか省略表現（`--sign --encrypt` → `-se` など）があるので参考にするとよい。また，詳しい使い方については [Manual (https://www.gnupg.org/documentation/manuals.html)](https://www.gnupg.org/documentation/manuals.html) を見たほうが参考になるかもしれない。