Développer sa bibliothèque de signature sur courbe ...

Arnaud Ebalard, Ryad Benadjila,
Jean-Pierre Flori, Karim Khalfallah
<prenom.nom@ssi.gouv.fr>

RUMP @ SSTIC2016

Déjà des fans !

KEvin @kevinhatry · 20 h

oula, "dev from scratch d'une lib de crypto elliptique" au #sstic alerte crapto ?

3 2

KEvin @kevinhatry · 22 h

visiblement @newsoft a traumatisé tous

Besoin

- Bibliothèque de signature sur courbe elliptique
- Différentes courbes et mécanismes de signatures :
  - pas qu'EC-DSA
  - pas que secp{256,385,521}r1
- Multi-cibles : μC (20KB RAM) + SoC/CPU
- Multi-arch : 16, 32, 64-bits
- Autonome (aucune dépendance externe)

Sécurité

leitmotiv

- C99 ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;
- indépendance vis-à-vis de la taille des mots machines ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;
- indépendance vis-à-vis de la taille des mots machines ;
- tests unitaires et de non-régression ;

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;
- indépendance vis-à-vis de la taille des mots machines ;
- tests unitaires et de non-régression ;
- vecteurs de test pour EC-{,KC,G,R,S,FS}DSA.

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;
- indépendance vis-à-vis de la taille des mots machines ;
- tests unitaires et de non-régression ;
- vecteurs de test pour EC-{,KC,G,R,S,FS}DSA.
- objectif de temps constant

Sécurité

leitmotiv

- C99 ;
- pas d'objectif de performance à tout prix ;
- allocation statique ;
- typage fort (dans les limites du langage C) ;
- pré et post conditions en entrée et sortie de fonction ;
- indépendance vis-à-vis de la taille des mots machines ;
- tests unitaires et de non-régression ;
- vecteurs de test pour EC-{,KC,G,R,S,FS}DSA.
- objectif de temps constant
- auditabilité (manuelle + outils)

```
$ ls *.c *.h
bitops.h      fp_add.c              sha384.h        ec_params.c
words_16.h    fp_add.h              sha512.c        ec_params.h
words_32.h    fp_config.h           sha512.h        ec_params_bplp224r1.h
words_64.h    fp_montgomery.h       sig_algs.h      ec_params_bplp256r1.h
words.h       fp_mul.c              ecdsa.c         ec_params_bplp384r1.h
nn_config.h   fp_mul.h              ecdsa.h         ec_params_bplp512r1.h
nn.h          fp_mul_redc1.c        ecfsdsa.c       ec_params_frp256v1.h
nn.c          fp_mul_redc1.h        ecfsdsa.h       ec_params_gost256.h
nn_add.c      fp_pow.c              ecgdsa.c        ec_params_gost512.h
nn_add.h      fp_pow.h              ecgdsa.h        ec_params_secp224r1.h
nn_div.c      ec_shortw_aff.h       eckcdsa.c       ec_params_secp256r1.h
nn_div.h      ec_shortw.h           eckcdsa.h       ec_params_secp384r1.h
nn_logical.c  ec_shortw_prj.c       ecosdsa.c       ec_params_secp521r1.h
nn_logical.h  ec_shortw_prj.h       ecosdsa.h       ec_self_tests.c
nn_modinv.c   ec_shortw_prj_monty.c ecrdsa.c        ec_self_tests.h
nn_modinv.h   ec_shortw_prj_monty.h ecrdsa.h        ec_tests.c
nn_mul.c      hash_algs.h           ecsdsa.c        ec_utils.c
nn_mul.h      sha224.c              ecsdsa_common.c tests.c
nn_rand.c     sha224.h              ecsdsa.h        utils.h
nn_rand.h     sha256.c              ec_key.c        rand.h
fp.h          sha256.h              ec_key.h        lib_ecc_types.h
fp.c          sha384.c              curves.h        lib_ecc_config.h
```

Types

```
$ ls *.c *.h
bitops.h      fp_add.c              sha384.h         ec_params.c
words_16.h    fp_add.h              sha512.c         ec_params.h
words_32.h    fp_config.h           sha512.h         ec_params_bplp224r1.h
words_64.h    fp_montgomery.h       sig_algs.h       ec_params_bplp256r1.h
words.h       fp_mul.c              ecdsa.c          ec_params_bplp384r1.h
nn_config.h   fp_mul.h              ecdsa.h          ec_params_bplp512r1.h
nn.h          fp_mul_redc1.c        ecfsdsa.c        ec_params_frp256v1.h
nn.c          fp_mul_redc1.h        ecfsdsa.h        ec_params_gost256.h
nn_add.c      fp_pow.c              ecgdsa.c         ec_params_gost512.h
nn_add.h      fp_pow.h              ecgdsa.h         ec_params_secp224r1.h
nn_div.c      ec_shortw_aff.h       eckcdsa.c        ec_params_secp256r1.h
nn_div.h      ec_shortw.h           eckcdsa.h        ec_params_secp384r1.h
nn_logical.c  ec_shortw_prj.c       ecosdsa.c        ec_params_secp521r1.h
nn_logical.h  ec_shortw_prj.h       ecosdsa.h        ec_self_tests.c
nn_modinv.c   ec_shortw_prj_monty.c ecrdsa.c         ec_self_tests.h
nn_modinv.h   ec_shortw_prj_monty.h ecrdsa.h         ec_tests.c
nn_mul.c      hash_algs.h           ecsdsa.c         ec_utils.c
nn_mul.h      sha224.c              ecsdsa_common.c  tests.c
nn_rand.c     sha224.h              ecsdsa.h         utils.h
nn_rand.h     sha256.c              ec_key.c         rand.h
fp.h          sha256.h              ec_key.h         lib_ecc_types.h
fp.c          sha384.c              curves.h         lib_ecc_config.h
```

Grands entiers **positifs** ($\mathbb{N}$, pas $\mathbb{Z}$)

```
$ ls *.c *.h
bitops.h      fp_add.c              sha384.h          ec_params.c
words_16.h    fp_add.h              sha512.c          ec_params.h
words_32.h    fp_config.h           sha512.h          ec_params_bplp224r1.h
words_64.h    fp_montgomery.h       sig_algs.h        ec_params_bplp256r1.h
words.h       fp_mul.c              ecdsa.c           ec_params_bplp384r1.h
nn_config.h   fp_mul.h              ecdsa.h           ec_params_bplp512r1.h
nn.h          fp_mul_redc1.c        ecfsdsa.c         ec_params_frp256v1.h
nn.c          fp_mul_redc1.h        ecfsdsa.h         ec_params_gost256.h
nn_add.c      fp_pow.c              ecgdsa.c          ec_params_gost512.h
nn_add.h      fp_pow.h              ecgdsa.h          ec_params_secp224r1.h
nn_div.c      ec_shortw_aff.h       eckcdsa.c         ec_params_secp256r1.h
nn_div.h      ec_shortw.h           eckcdsa.h         ec_params_secp384r1.h
nn_logical.c  ec_shortw_prj.c       ecosdsa.c         ec_params_secp521r1.h
nn_logical.h  ec_shortw_prj.h       ecosdsa.h         ec_self_tests.c
nn_modinv.c   ec_shortw_prj_monty.c ecrdsa.c          ec_self_tests.h
nn_modinv.h   ec_shortw_prj_monty.h ecrdsa.h          ec_tests.c
nn_mul.c      hash_algs.h           ecsdsa.c          ec_utils.c
nn_mul.h      sha224.c              ecsdsa_common.c   tests.c
nn_rand.c     sha224.h              ecsdsa.h          utils.h
nn_rand.h     sha256.c              ec_key.c          rand.h
fp.h          sha256.h              ec_key.h          lib_ecc_types.h
fp.c          sha384.c              curves.h          lib_ecc_config.h
```

$\mathbb{F}_p$ avec p premier (i.e. pas de $\mathbb{F}_{2^m}$)

```
$ ls *.c *.h
bitops.h        fp_add.c              sha384.h          ec_params.c
words_16.h      fp_add.h              sha512.c          ec_params.h
words_32.h      fp_config.h           sha512.h          ec_params_bplp224r1.h
words_64.h      fp_montgomery.h       sig_algs.h        ec_params_bplp256r1.h
words.h         fp_mul.c              ecdsa.c           ec_params_bplp384r1.h
nn_config.h     fp_mul.h              ecdsa.h           ec_params_bplp512r1.h
nn.h            fp_mul_redc1.c        ecfsdsa.c         ec_params_frp256v1.h
nn.c            fp_mul_redc1.h        ecfsdsa.h         ec_params_gost256.h
nn_add.c        fp_pow.c              ecgdsa.c          ec_params_gost512.h
nn_add.h        fp_pow.h              ecgdsa.h          ec_params_secp224r1.h
nn_div.c        ec_shortw_aff.h       eckcdsa.c         ec_params_secp256r1.h
nn_div.h        ec_shortw.h           eckcdsa.h         ec_params_secp384r1.h
nn_logical.c    ec_shortw_prj.c       ecosdsa.c         ec_params_secp521r1.h
nn_logical.h    ec_shortw_prj.h       ecosdsa.h         ec_self_tests.c
nn_modinv.c     ec_shortw_prj_monty.c ecrdsa.c          ec_self_tests.h
nn_modinv.h     ec_shortw_prj_monty.h ecrdsa.h          ec_tests.c
nn_mul.c        hash_algs.h           ecsdsa.c          ec_utils.c
nn_mul.h        sha224.c              ecsdsa_common.c   tests.c
nn_rand.c       sha224.h              ecsdsa.h          utils.h
nn_rand.h       sha256.c              ec_key.c          rand.h
fp.h            sha256.h              ec_key.h          lib_ecc_types.h
fp.c            sha384.c              curves.h          lib_ecc_config.h
```

Courbes (sur $\mathbb{F}_p$ uniquement)

```
$ ls *.c *.h
bitops.h        fp_add.c              sha384.h          ec_params.c
words_16.h      fp_add.h              sha512.c          ec_params.h
words_32.h      fp_config.h           sha512.h          ec_params_bplp224r1.h
words_64.h      fp_montgomery.h       sig_algs.h        ec_params_bplp256r1.h
words.h         fp_mul.c              ecdsa.c           ec_params_bplp384r1.h
nn_config.h     fp_mul.h              ecdsa.h           ec_params_bplp512r1.h
nn.h            fp_mul_redc1.c        ecfsdsa.c         ec_params_frp256v1.h
nn.c            fp_mul_redc1.h        ecfsdsa.h         ec_params_gost256.h
nn_add.c        fp_pow.c              ecgdsa.c          ec_params_gost512.h
nn_add.h        fp_pow.h              ecgdsa.h          ec_params_secp224r1.h
nn_div.c        ec_shortw_aff.h       eckcdsa.c         ec_params_secp256r1.h
nn_div.h        ec_shortw.h           eckcdsa.h         ec_params_secp384r1.h
nn_logical.c    ec_shortw_prj.c       ecosdsa.c         ec_params_secp521r1.h
nn_logical.h    ec_shortw_prj.h       ecosdsa.h         ec_self_tests.c
nn_modinv.c     ec_shortw_prj_monty.c ecrdsa.c          ec_self_tests.h
nn_modinv.h     ec_shortw_prj_monty.h ecrdsa.h          ec_tests.c
nn_mul.c        hash_algs.h           ecsdsa.c          ec_utils.c
nn_mul.h        sha224.c              ecsdsa_common.c   tests.c
nn_rand.c       sha224.h              ecsdsa.h          utils.h
nn_rand.h       sha256.c              ec_key.c          rand.h
fp.h            sha256.h              ec_key.h          lib_ecc_types.h
fp.c            sha384.c              curves.h          lib_ecc_config.h
```

SHA-{224,256,384,512}

```
$ ls *.c *.h
bitops.h        fp_add.c              sha384.h          ec_params.c
words_16.h      fp_add.h              sha512.c          ec_params.h
words_32.h      fp_config.h           sha512.h          ec_params_bplp224r1.h
words_64.h      fp_montgomery.h       sig_algs.h        ec_params_bplp256r1.h
words.h         fp_mul.c              ecdsa.c           ec_params_bplp384r1.h
nn_config.h     fp_mul.h              ecdsa.h           ec_params_bplp512r1.h
nn.h            fp_mul_redc1.c        ecfsdsa.c         ec_params_frp256v1.h
nn.c            fp_mul_redc1.h        ecfsdsa.h         ec_params_gost256.h
nn_add.c        fp_pow.c              ecgdsa.c          ec_params_gost512.h
nn_add.h        fp_pow.h              ecgdsa.h          ec_params_secp224r1.h
nn_div.c        ec_shortw_aff.h       eckcdsa.c         ec_params_secp256r1.h
nn_div.h        ec_shortw.h           eckcdsa.h         ec_params_secp384r1.h
nn_logical.c    ec_shortw_prj.c       ecosdsa.c         ec_params_secp521r1.h
nn_logical.h    ec_shortw_prj.h       ecosdsa.h         ec_self_tests.c
nn_modinv.c     ec_shortw_prj_monty.c ecrdsa.c          ec_self_tests.h
nn_modinv.h     ec_shortw_prj_monty.h ecrdsa.h          ec_tests.c
nn_mul.c        hash_algs.h           ecsdsa.c          ec_utils.c
nn_mul.h        sha224.c              ecsdsa_common.c   tests.c
nn_rand.c       sha224.h              ecsdsa.h          utils.h
nn_rand.h       sha256.c              ec_key.c          rand.h
fp.h            sha256.h              ec_key.h          lib_ecc_types.h
fp.c            sha384.c              curves.h          lib_ecc_config.h
```

EC-{,CK,G,R,S,FS}DSA

```
$ ls *.c *.h
bitops.h        fp_add.c                sha384.h        ec_params.c
words_16.h      fp_add.h                sha512.c        ec_params.h
words_32.h      fp_config.h             sha512.h        ec_params_bplp224r1.h
words_64.h      fp_montgomery.h         sig_algs.h      ec_params_bplp256r1.h
words.h         fp_mul.c                ecdsa.c         ec_params_bplp384r1.h
nn_config.h     fp_mul.h                ecdsa.h         ec_params_bplp512r1.h
nn.h            fp_mul_redc1.c          ecfsdsa.c       ec_params_frp256v1.h
nn.c            fp_mul_redc1.h          ecfsdsa.h       ec_params_gost256.h
nn_add.c        fp_pow.c                ecgdsa.c        ec_params_gost512.h
nn_add.h        fp_pow.h                ecgdsa.h        ec_params_secp224r1.h
nn_div.c        ec_shortw_aff.h         eckcdsa.c       ec_params_secp256r1.h
nn_div.h        ec_shortw.h             eckcdsa.h       ec_params_secp384r1.h
nn_logical.c    ec_shortw_prj.c         ecosdsa.c       ec_params_secp521r1.h
nn_logical.h    ec_shortw_prj.h         ecosdsa.h       ec_self_tests.c
nn_modinv.c     ec_shortw_prj_monty.c   ecrdsa.c        ec_self_tests.h
nn_modinv.h     ec_shortw_prj_monty.h   ecrdsa.h        ec_tests.c
nn_mul.c        hash_algs.h             ecsdsa.c        ec_utils.c
nn_mul.h        sha224.c                ecsdsa_common.c tests.c
nn_rand.c       sha224.h                ecsdsa.h        utils.h
nn_rand.h       sha256.c                ec_key.c        rand.h
fp.h            sha256.h                ec_key.h        lib_ecc_types.h
fp.c            sha384.c                curves.h        lib_ecc_config.h
```

Courbes secp*r1, brainpoolP*r1, FRP256v1

```
$ ls *.c *.h
bitops.h        fp_add.c                sha384.h          ec_params.c
words_16.h      fp_add.h                sha512.c          ec_params.h
words_32.h      fp_config.h             sha512.h          ec_params_bplp224r1.h
words_64.h      fp_montgomery.h         sig_algs.h        ec_params_bplp256r1.h
words.h         fp_mul.c                ecdsa.c           ec_params_bplp384r1.h
nn_config.h     fp_mul.h                ecdsa.h           ec_params_bplp512r1.h
nn.h            fp_mul_redc1.c          ecfsdsa.c         ec_params_frp256v1.h
nn.c            fp_mul_redc1.h          ecfsdsa.h         ec_params_gost256.h
nn_add.c        fp_pow.c                ecgdsa.c          ec_params_gost512.h
nn_add.h        fp_pow.h                ecgdsa.h          ec_params_secp224r1.h
nn_div.c        ec_shortw_aff.h         eckcdsa.c         ec_params_secp256r1.h
nn_div.h        ec_shortw.h             eckcdsa.h         ec_params_secp384r1.h
nn_logical.c    ec_shortw_prj.c         ecosdsa.c         ec_params_secp521r1.h
nn_logical.h    ec_shortw_prj.h         ecosdsa.h         ec_self_tests.c
nn_modinv.c     ec_shortw_prj_monty.c   ecrdsa.c         ec_self_tests.h
nn_modinv.h     ec_shortw_prj_monty.h   ecrdsa.h         ec_tests.c
nn_mul.c        hash_algs.h             ecsdsa.c          ec_utils.c
nn_mul.h        sha224.c                ecsdsa_common.c   tests.c
nn_rand.c       sha224.h                ecsdsa.h          utils.h
nn_rand.h       sha256.c                ec_key.c          rand.h
fp.h            sha256.h                ec_key.h          lib_ecc_types.h
fp.c            sha384.c                curves.h          lib_ecc_config.h
```

tests unitaires + vecteurs

Configuration

$ cat lib_ecc_config.h

```
#ifndef __LIB_ECC_CONFIG_H__
#define __LIB_ECC_CONFIG_H__

/*
 * This configuration file provides
 * various knobs to configure what
 * will be built in the library
 */

/* Supported curves */
#define WITH_CURVE_FRP256V1
#define WITH_CURVE_SECP224R1
#define WITH_CURVE_SECP256R1
#define WITH_CURVE_SECP384R1
#define WITH_CURVE_SECP521R1
#define WITH_CURVE_BRAINPOOLP224R1
#define WITH_CURVE_BRAINPOOLP256R1
#define WITH_CURVE_BRAINPOOLP384R1
#define WITH_CURVE_BRAINPOOLP512R1
#define WITH_CURVE_GOST256
#define WITH_CURVE_GOST512

/* Supported hash algorithms */
#define WITH_HASH_SHA224
#define WITH_HASH_SHA256
#define WITH_HASH_SHA384
#define WITH_HASH_SHA512

/* Supported sig/verif schemes */
#define WITH_SIG_ECDSA
#define WITH_SIG_ECKCDSA
#define WITH_SIG_ECSDSA
#define WITH_SIG_ECOSDSA
#define WITH_SIG_ECFSDSA
#define WITH_SIG_ECGDSA
#define WITH_SIG_ECRDSA

#endif /* __LIB_ECC_CONFIG_H__ */
```

Publication

- Double licence BSD / GPLv2+
- Bientôt i.e. après ...
- ... Stabilisation API de signature
- ... Finalisation analyse (manuelle + outils)
- ... Finalisation temps constant
- ... Doc (humm humm)

Un mot sur les normes . . .

Utiliser les normes nationales ?

Utiliser les normes nationales ?

6.4.1. 서명 생성 과정

비공개 서명 키 d를 가진 서명자의 메시지 M에 대한 서명은 다음과 같은 과정을 통하여 생성된 두 개의 바이트 열 r과 s의 쌍으로 구성된다.

입력 : 서명할 메시지 M, 서명자의 타원 곡선 도메인 변수 및 비공개 서명 키·공개 검증 키 쌍.

출력 : M에 대한 서명으로 바이트 열 $\{r, s\}$.

과정 :

단계 1. 난수값 k를 $\{1, 2, \cdots, n-1\}$에서 생성한다.

단계 2. 타원 곡선의 점 $(x_1, y_1) = kG$를 계산한다. 여기서 유한체 원소 x_1, y_1은 바이트 열로 변환된 형태이다.

단계 3. 서명의 첫 부분 $r = H(x_1)$을 계산한다.

단계 4. 메시지의 해시 코드 $v = H(c_Q \| M)$을 계산한다.

단계 5. 중간값 $e = r \oplus v \bmod n$을 계산한다.

단계 6. $t = d(k-e) \bmod n$을 계산한다. 만약 $t = 0$이면 단계 1에서부터 다시 수행

Utiliser les normes nationales ?

Security by obscurity ?

6.4.1. 서명 생성 과정

비공개 서명 키 d를 가진 서명자의 메시지 M에 대한 서명은 다음과 같은 과정을 통하여 생성된 두 개의 바이트 열 r과 s의 쌍으로 구성된다.

입력 : 서명할 메시지 M, 서명자의 타원 곡선 도메인 변수 및 비공개 서명 키·공개 검증 키 쌍.

출력 : M에 대한 서명으로 바이트 열 $\{r, s\}$.

과정 :

단계 1. 난수값 k를 $\{1, 2, \cdots, n-1\}$에서 생성한다.

단계 2. 타원 곡선의 점 $(x_1, y_1) = kG$를 계산한다. 여기서 유한체 원소 x_1, y_1은 바이트 열로 변환된 형태이다.

단계 3. 서명의 첫 부분 $r = H(x_1)$을 계산한다.

단계 4. 메시지의 해시 코드 $v = H(c_Q \| M)$을 계산한다.

단계 5. 중간값 $e = r \oplus v \bmod n$을 계산한다.

단계 6. $t = d(k-e) \bmod n$을 계산한다. 만약 $t = 0$이면 단계 1에서부터 다시 수행

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

> k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

6.4 ECDSA Digital Signature Generation and Verification

An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

1. Domain parameters that are generated in accordance with Section 6.1.1,
2. A private key that is generated as specified in Section 6.2.1,
3. A per-message secret number that is generated as specified in Section 6.3,

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

  k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

  6.4 ECDSA Digital Signature Generation and Verification

  An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

  1. Domain parameters that are generated in accordance with Section 6.1.1,
  2. A private key that is generated as specified in Section 6.2.1,
  3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

  k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

  6.4 ECDSA Digital Signature Generation and Verification

  An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

  1. Domain parameters that are generated in accordance with Section 6.1.1,
  2. A private key that is generated as specified in Section 6.2.1,
  3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

  k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

  6.4 ECDSA Digital Signature Generation and Verification

  An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

  1. Domain parameters that are generated in accordance with Section 6.1.1,
  2. A private key that is generated as specified in Section 6.2.1,
  3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~ 2005

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

  k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

  6.4 ECDSA Digital Signature Generation and Verification

  An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

  1. Domain parameters that are generated in accordance with Section 6.1.1,
  2. A private key that is generated as specified in Section 6.2.1,
  3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~ 2005
- ISO 14888-3

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

  k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.

  6.4 ECDSA Digital Signature Generation and Verification

  An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:

  1. Domain parameters that are generated in accordance with Section 6.1.1,
  2. A private key that is generated as specified in Section 6.2.1,
  3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~ 2005
- ISO 14888-3 ~~2006~~

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

> k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.
>
> **6.4 ECDSA Digital Signature Generation and Verification**
>
> An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:
>
> 1. Domain parameters that are generated in accordance with Section 6.1.1,
> 2. A private key that is generated as specified in Section 6.2.1,
> 3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~ 2005
- ISO 14888-3 ~~2006~~ 2016

Quelle norme ?

- IETF (RFC4753,5639,5903,6932,6954,6989, . . .) ?
- FIPS 186-4 (DSS)

> k and k^{-1} may be pre-computed, since knowledge of the message to be signed is not required for the computations. When k and k^{-1} are pre-computed, their confidentiality and integrity **shall** be protected.
>
> **6.4 ECDSA Digital Signature Generation and Verification**
>
> An ECDSA digital signature (r, s) **shall** be generated as specified in ANS X9.62, using:
>
> 1. Domain parameters that are generated in accordance with Section 6.1.1,
> 2. A private key that is generated as specified in Section 6.2.1,
> 3. A per-message secret number that is generated as specified in Section 6.3,

- ANSI X9.62 ~~1998~~ 2005
- ISO 14888-3 ~~2006~~ 2016
- ...

Wait a sec ...

Signature EC-KCDSA

Jeu des différences

Calcul de R en 6.7.1

$R = h(FE2BS(r, \Pi_X))$.

If γ is longer than β, then the witness function is replaced by the formula

$R = I2BS(\beta, BS2I(\gamma, h(FE2BS(r, \Pi_X))) \bmod 2\beta)$.

Calcul de R en 6.7.4.4

The signing entity computes $R = h(FE2BS(r, \Pi_X))$, where output of h is the hash-code of Π_X. If γ is longer than β, then the computation of witness is replaced by $R = I2BS(\beta, BS2I(8 \lceil \log_{256}(r) \rceil, FE2BS(r, \Pi_X)) \bmod 2\beta)$.

Signature EC-RDSA ISO 14888-3 : 2016

Génération du *randomizer* K

The signing entity generates a random or pseudo-random integer K such that $0 < K < Q$.

Vecteurs de test dans la norme

Q = 4531ACD1 FE0023C7 550D267B 6B2FEE80 922B14B2 FFB90F04
D4EB7C09 B5D2D15D A82F2D7E CB1DBAC7 19905C5E ECC423F1
D86E25ED BE23C595 D644AAF1 87E6E6DF

K = C573F6B3 01D99C24 C422A427 1E9EC93B AEAA6EEF 0DE82477
D8B7391F 9F6790D9 DDE5146F 02ECA567 2C38FC80 9CF4CA88
937C4B3A 3936ADF9 908F796C 86C05C43

Signature EC-RDSA ISO 14888-3 : 2016

Génération du *randomizer* K

The signing entity generates a random or pseudo-random integer K such that $0 < K < Q$.

Vecteurs de test dans la norme

Q = 4531ACD1 FE0023C7 550D267B 6B2FEE80 922B14B2 FFB90F04
D4EB7C09 B5D2D15D A82F2D7E CB1DBAC7 19905C5E ECC423F1
D86E25ED BE23C595 D644AAF1 87E6E6DF

K = C573F6B3 01D99C24 C422A427 1E9EC93B AEAA6EEF 0DE82477
D8B7391F 9F6790D9 DDE5146F 02ECA567 2C38FC80 9CF4CA88
937C4B3A 3936ADF9 908F796C 86C05C43

Du coup . . .

Notre lib n'est en pratique **pas encore totalement compatible** avec l'ISO 14888-3 :2016 mais on travaille

. . .

Du coup ...

Notre lib n'est en pratique **pas encore totalement compatible** avec l'ISO 14888-3:2016 mais on travaille ...**à corriger la norme.**

Q ?