Tepla

吉田 努

白勢研ゼミ 2016/06/07

前回

- 類別
- Tepla

今回

- Teplaについて詳しく

Tepla

- 筑波大学が作成したペアリング演算ライブラリ
- C言語
- GMPとOpenSSLを必要とする
- 提供する機能
  - 有限体上の演算
  - 楕円曲線の演算
  - ペアリング演算
- 最近version 2がリリースされた

GMP

- 任意精度演算
  - 要するに巨大な桁数が簡単に計算できる
- 自分で作るのは難しい

なにができそうか

- ペアリングを使った暗号の構築
  - IDベース暗号
  - タイムリリース暗号
  - プロキシ暗号
- 楕円曲線暗号系の実装
  - 使える楕円曲線が決まっている
    - 問題にはならない？
  - 余り面白くないかもしれない -
  - GMPを使って実装するよりは楽？
    - 楕円曲線の演算を自作する必要無し
    - APIを呼ぶだけ

ペアリング

- $E[n] \times E[n] \to \mu_n$
- 暗号で使うのは双線形性
- 公開鍵暗号を拡張したような暗号を構成できる

TEPLAの仕様

- オブジェクト指向っぽい作り
  - 構造体と関数へのポインタで実現
  - 内部では,関数?メソッド？は->(アロー演算子)で呼ぶ
  - 外部(ユーザが使う部分)では普通の関数呼び出し
- ユーザが使うAPIはドキュメント付き
  - 素晴らしい!
  - 日本語/英語

一応Cの復習

- 構造体
  - 複数のデータを持つことができる変数？
  - 構造体の中身の変数をメンバ変数と言う
  - a->b; aという構造体のメンバ変数であるb
- ポインタ
  - オブジェクト？(変数？)のアドレスを指す変数

抜粋 from README

- tepla
  - ec_lib.c : interfaces of operation function.
  - hash.c : interfaces of hash function.
- ec_bn254
  - bn254_fp.c : prime field.
  - bn254_fp2.c : quadratic extension field.
  - bn254_fp6.c : sextic extension field.
  - bn254_fp12.c : twelvetic extension field.
  - ec_bn254_fp.c : elliptic curve over prime field.
  - ec_bn254_fp2.c : twisted elliptic curve over quadratic extension field.
  - ec_bn254_pairing.c : pairing function.
  - ec_bn254_lib.c : functions to create a instance of pairing structure.

一部抜粋 from ec_lib.c

```
void curve_init(EC_GROUP ec, const char *param)
{
    if (strcmp(param, "ec_bn254_fpa") == 0)
    {
        ec->curve_init = ec_bn254_fpa_group_new;
        ec->curve_clear = ec_bn254_group_clear;
    }
    else if (strcmp(param, "ec_bn254_twa") == 0)
    {
        ec->curve_init = ec_bn254_twa_group_new;
        ec->curve_clear = ec_bn254_group_clear;
    }
    else if (strcmp(param, "ec_bn254_fpb") == 0)
    {
        ec->curve_init = ec_bn254_fpb_group_new;
        ec->curve_clear = ec_bn254_group_clear;
    }
    else if (strcmp(param, "ec_bn254_twb") == 0)
    {
        ec->curve_init = ec_bn254_twb_group_new;
        ec->curve_clear = ec_bn254_group_clear;
    }
    else
    {
```

```
}
```

一部抜粋 from ec_lib.c

```
void pairing_map(Element g, const EC_POINT P, const EC_POINT Q, const EC_
{
    p->pairing(g, Q, P, p);
}
```

- アロー演算子で関数を呼ぶ
  - つまりEC_PAIRING構造体はpairingという関数へのポインタを持つはず

構造体を見ると

```
typedef struct ec_pairing_st
{
    PairingType type;

    char* pairing_name;

    void (*pairing)(Element z, const EC_POINT x, const EC_POINT y, const
    void (*pairing_double)(Element z, const EC_POINT x1, const EC_POINT y

    EC_GROUP g1;
    EC_GROUP g2;

    Field    g3;

    void* precomp;

} EC_PAIRING[1];
```

- 関数へのポインタがあるが分かった
- ではどこを参照しているのか？

Cのテクニック

```
    void* precomp;

} EC_PAIRING[1]
```

- typedef struct ec_pairing_st{...}EC_PAIRING[1];
- 一個の配列としてtypedef
- ポインタでは恐らく駄目
- メリットは変数を渡しやすくなる？

from ec_bn254_lib.c

```
void ec_bn254_pairing_a_new(EC_PAIRING p)
{
    p->type = Pairing_ECBN254a;

    set_pairing_name(p, "ECBN254a");

    p->pairing = ec_bn254_pairing_beuchat;
    p->pairing_double = ec_bn254_double_pairing_beuchat;

    curve_init(p->g1, "ec_bn254_fpa");
    curve_init(p->g2, "ec_bn254_twa");

    field_init(p->g3, "bn254_fp12a");

    ec_bn254_pairing_precomp_beuchat(p);
}
```

- 結局のところ"ec_bn254_pairing_beuchat"とやらを呼んでいるらしい
- インスタンス生成時にpairing変数に関数を指定している
- 多分コンストラクタに相当？

TEPLAの構造 from README

- tepla
  - ec_lib.c : interfaces of operation function.
  - hash.c : interfaces of hash function.
- ec_bn254
  - bn254_fp.c : prime field.
  - bn254_fp2.c : quadratic extension field.
  - bn254_fp6.c : sextic extension field.
  - bn254_fp12.c : twelvetic extension field.
  - ec_bn254_fp.c : elliptic curve over prime field.
  - ec_bn254_fp2.c : twisted elliptic curve over quadratic extension field.
  - ec_bn254_pairing.c : pairing function.
  - ec_bn254_lib.c : functions to create a instance of pairing structure.

Documentを見る

sampleを見る

まとめ

- Teplaを使うに当たって
  - ドキュメントを読む
    - 結構親切
- Teplaをインストールする
  - linux推奨