

ペアリングの応用

光成滋生

last update: 2025/11/04

概要

目的

- ペアリングの性質, BLS署名の理解
- 秘密分散
- DKG

ペアリング

type-1ペアリング (Weil pairing)

- E/\mathbb{F}_p : 楕円曲線, $G_1 := \langle P_1 \rangle = \{0, P_1, 2P_1, \dots, (r-1)P_1\}$: 位数 r の加法巡回群
乗法巡回群表記のテキストもある
- $G_T := \langle g \rangle := \{1, g, g^2, \dots, g^{r-1}\} \subseteq \mathbb{F}_{p^k}$: 1の r 乗根 g からなる乗法巡回群
- $e : G_1 \times G_1 \rightarrow G_T$ がペアリングとは $e(aP_1, bP_1) = g^{ab}$ を満たすもの

双線型性

- $P, Q, R \in G_1$ に対して $P = aP_1, Q = bP_1, R = cP_1$ とすると
 - $e(P + Q, R) = e(P, R)e(Q, R) = g^{(a+b)c}$
 - $e(P, Q + R) = e(P, Q)e(P, R) = g^{a(b+c)}$
 - G_T も加法群とみなすと通常の $f(x + y) = f(x) + f(y)$ の線型性の形
 - 2変数の両方に関して線型なので双線型

ECDLPとペアリングの関係

もともとはECDLPを解くために利用された

- $P, aP \in G_1$ が与えられたときに a を求めたい
 - $g = e(P, P), e(P, aP) = g^a$ なので g, g^a に関するDLPが解ければECDLPも解ける
 - MOV (Menezes, Okamoto, Vanstone) リダクションという

3人の間の鍵共有 (Joux, 2000)

- A, B, C がそれぞれ秘密鍵 $a, b, c \in \mathbb{F}_r$ を持ち aP, bP, cP を共有する
- A は $e(bP, cP)^a$, B は $e(cP, aP)^b$, C は $e(aP, bP)^c$ を計算する
 - それぞれ g^{abc} になるので鍵共有ができた
- より多数の鍵共有ができるか（多重線型写像の構成）は未解決（否定的な結果が優勢）

主な安全性仮定の問題

- CBDH (Computational Bilinear DH): (P, aP, bP, cP) に対して $e(P, P)^{abc}$ を計算する
- DBDH (Decisional BDH): (P, aP, bP, cP, g') に対して $g' = e(P, P)^{abc}$ を判定する

BLS署名

CBDH仮定の元で安全な署名

- Boneh, Lynn, Shacham (2001)
 - $H : \{0, 1\}^* \rightarrow G_1$ をハッシュ関数
- KeyGen: 署名鍵: $s \xleftarrow{U} \mathbb{F}_r$, 検証鍵 $P := sP_1$
- Sign: メッセージ m に対して $\sigma := sH(m)$
- Verify: $e(H(m), P) = e(\sigma, P_1)$ なら valid
 - 正当性: $\text{LHS} = e(H(m), sP_1) = e(sH(m), P_1) = \text{RHS}$

特徴

- 署名長が短い (G_1 の要素1個) , 乱数不要の決定的アルゴリズム

type-3ペアリング

ペアリングの重要性と高速化の必要性

- ペアリングの重要性に伴いより演算効率のよい写像が望まれている
- 現在の主流は非対称ペアリング
 $e : G_1 \times G_2 \rightarrow G_T$: で $G_1 \neq G_2$ かつ G_1 と G_2 の間に効率的な同型写像が存在しない

BLS12-381曲線 (Barreto, Lynn, Scott)

- BLS写像のBLSとは無関係
- $E/\mathbb{F}_p: y^2 = x^3 + 4$ (p : 381bit素数)
- 写像先の $G_T \subseteq \mathbb{F}_{p^{12}}$ は \mathbb{F}_p の $k = 12$ 次拡大体 (BLS12の12)
- $G_1 := \langle P_1 \rangle \subseteq E(\mathbb{F}_p)$: P_1 の位数は255bit素数 r
- $G_2 := \langle P_2 \rangle \subseteq E'(\mathbb{F}_{p^2})$: 2次拡大体上の楕円曲線 $E' : y^2 = x^3 + 4(1 + i), i^2 = -1$
- $G_T = \langle g \rangle = \langle e(P_1, P_2) \rangle$
 - $e(aP_1, bP_2) = e(P_1, P_2)^{ab} = g^{ab}$ を満たす

type-3ペアリングでのBLS署名

type-1版をtype-3版に適用する

- $G_i := \langle P_i \rangle, H_i : \{0, 1\}^* \rightarrow G_i$ ($i = 1, 2$): ハッシュ関数, $b \in \{1, 2\}$ を固定
- 署名鍵: $s \xleftarrow{U} \mathbb{F}_r$ に対して
 - $b = 1$ のとき 検証鍵 $P := sP_1 \in G_1$, 署名 $\sigma := sH_2(m) \in G_2$
 - 検証: $e(P, H_2(m)) = e(P_1, \sigma)$ なら valid
 - $b = 2$ のとき 検証鍵 $P := sP_2 \in G_2$, 署名 $\sigma := sH_1(m) \in G_1$
 - 検証: $e(H_1(m), P) = e(\sigma, P_2)$ なら valid
- 検証鍵が小さい方がよいなら $b = 1$, 署名が小さい方がよいなら $b = 2$
 - Ethereumは $b = 1$ (検証鍵が G_1 の要素) を採用 (以下では $b = 1$ とする)

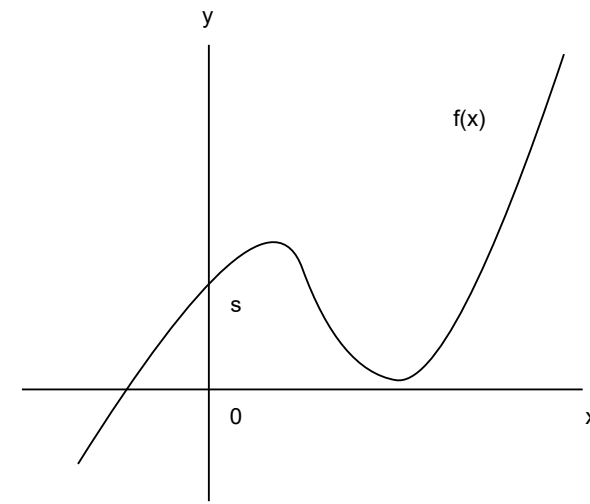
安全性仮定

- co-CDH仮定: $P_1, aP_1 \in G_1, P_2 \in G_2$ に対して aP_2 が計算困難 ($b = 2$ も同様)

Shamirの秘密分散法

(t, n) 閾値秘密分散 SSS (Shamir's Secret Sharing)

- K : 有限体, 秘密 $s \in K$ を n 個のシェア $s_1, \dots, s_n \in K$ に分割
- 任意の t 個のシェアで秘密を再構成可能



Lagrange補間による構成法

- $a_1, \dots, a_{t-1} \in K$: ランダム, $a_0 := s$, $f(x) := \sum_{i=0}^{t-1} a_i x^i$
ディラーはユーザ $i \in [1, n]$ に秘密のシェア $s_i := f(i)$ を渡す. みんなの ID: $u_i \in K^*$ は共有
- n 人のうち t 人の $S := \{u_{i_1}, \dots, u_{i_t}\}$ が集まりシェアを共有
 - $u \in S$ に対して $\Delta_{u,S}(x) := \prod_{v \in S \setminus u} \frac{x-v}{u-v}$ は $t-1$ 次多項式
 $\Delta_{u,S}(u) = 1, \Delta_{u,S}(w) = 0$ ($w \in S \setminus u$) なので $g(x) := \sum_{j=1}^t s_{u_j} \Delta_{u_j,S}(x)$
は $g(u_j) = s_{u_j}$ となり $f(x)$ に一致する ($t-1$ 次曲線は t 個の点で一意に定まる)
- 特に $f(0) = s$ が求まる
- $t-1$ 以下の人数が集まっても $f(0) \in K$ は不定なので情報量的安全性を持つ

(2, 3) 閾値秘密分散

- 秘密 $s = 12 \in \mathbb{F}_{257}$, $k = 2$ なので $s_1 = 100$ を選び1次多項式を $f(x) = s + s_1x$ とする
- ユーザIDは1, 2, 3とする. シェアは $s_1 = f(1) = 112$, $s_2 = f(2) = 212$, $s_3 = f(3) = 55$

シェアの再構成

- ユーザ1と3が集まった場合: $S = \{1, 3\}$
 - $\Delta_{1,\{1,3\}}(x) = \frac{x-3}{1-3} = -x/2 + 3/2$, $\Delta_{3,\{1,3\}}(x) = \frac{x-1}{3-1} = x/2 - 1/2$
 - $f(x) = 112(-x/2 + 3/2) + 55(x/2 - 1/2) = (57/2)x + (281/2)$
 - $f(0) = 281/2 \equiv 12 \pmod{257}$ が得られる

応用例

- 社長の秘密鍵を取締役員でシェア
 - 複数の取締役員の賛成で秘密鍵を復元

BLS署名への秘密分散の適用

BLS署名の構成

- $G_1 := \langle P_0 \rangle, G_2 := \langle Q \rangle, H : \{0, 1\}^* \rightarrow G_2$: ハッシュ関数とする
- 署名鍵 $s \in \mathbb{F}_r$ を (t, n) 閾値秘密分散で分割しシェア s_i をユーザID u_i に配付

ユーザ i の署名・検証

- 署名鍵: s_i , 検証鍵 $P_i := s_i P \in G_1$
- メッセージ m に対する署名: $\sigma_i := s_i H(m) \in G_2$
- 検証: $e(P_i, H(m)) = e(P_0, \sigma_i)$ なら valid

署名の復元

- t 人のユーザの署名 $\{\sigma_{i_j}\}$ から $\sigma := \sum_{j=1}^t \sigma_{i_j} \Delta_{u_{i_j}, s}(0) H(m) = s H(m)$ が得られる
 - 検証鍵と署名の両方が「署名鍵 $\times G_i$ の元」の形なのでLagrange補間が可能
- 全体の検証鍵 $P = s P_0$ で検証可能: $e(P, H(m)) = e(P_0, \sigma)$ なら valid
- 単なる秘密分散と異なり署名は**繰り返し利用可能**: 多数決にも利用できる

集約署名

複数の署名の集約

- ユーザ i の署名鍵 $s_i \in \mathbb{F}_r$, 検証鍵 $P_i := s_i P_0 \in G_1$
- 集約検証鍵: $P := P_1 + \dots + P_n = sP_0$ ($s := s_1 + \dots + s_n$ は誰も知らない)
- メッセージ m に対するユーザの署名: $\sigma_i := s_i H(m) \in G_2$
 - 各ユーザの署名の検証: $e(P_i, H(m)) = e(P_0, \sigma_i)$ なら valid
- 集約署名: $\sigma := \sigma_1 + \dots + \sigma_n = sH(m)$, $e(P, H(m)) = e(P_0, \sigma)$ なら valid

Rogue key攻撃 (ユーザ n が不正な (rogue) 攻撃者とする)

- 検証鍵: $P_n := aP_0 - (P_1 + \dots + P_{n-1})$, 署名: $\sigma_n := aH(m) - (\sigma_1 + \dots + \sigma_{n-1})$
- 集約検証鍵: $P = aP_0$, 集約署名: $\sigma = aH(m)$ は検証を通過してしまう (a は自由に選ぶ)
- 対策: PoP (Proof of Possession)
 - 事前に各ユーザは自身の検証鍵を自身の署名鍵で署名してみなが検証しておく
 - 攻撃者は自身の署名鍵を知らないので署名できない

ディーラーの不正対策

問題点

- 秘密分散のディーラーに権限集中
 - 秘密情報 s に対して $f(x) = s + s_1x + \dots + s_{t-1}x^{t-1}$ を作り $f(u_i)$ を配付
 - 各ユーザの $f(u_i)$ を知っている/不正なシェア配付の可能性

検証可能な秘密分散VSS (Verifiable Secret Sharing)

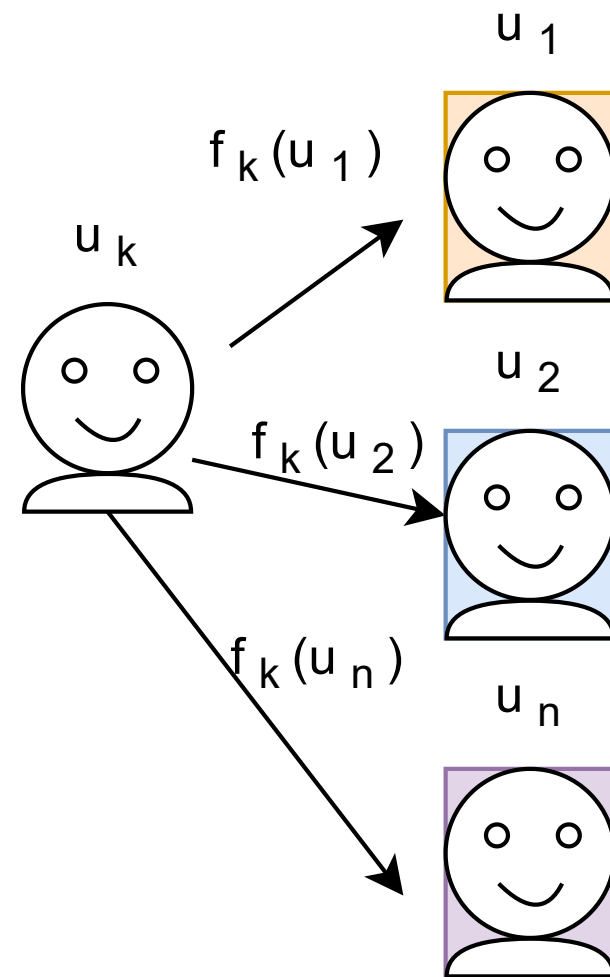
- 各ユーザが受け取ったシェアが正しいことを各自が検証可能にする
- ディーラーはシェア配付時に $\{Q_j\} = \{sP_0, s_1P_0, \dots, s_{t-1}P_0\}$ を公開
- 各ユーザは $sP_0 + u_i(s_1P_0) + \dots + u_i^{t-1}(s_{t-1}P_0) \stackrel{?}{=} f(u_i)P_0$ を検証

権限集中の対策は？

分散鍵生成 DKG (Distributed Key Generation)

全員がディーラーになる

- ユーザID $u_1, \dots, u_n (\neq 0)$ は既知
- ユーザ k が $f_k(x) = \sum_{i=0}^{t-1} s_{k,i} x^i$ を作り
ユーザ j に $f_k(u_j)$ を秘密裏に配付, $\{s_{k,i} P_0\}_{i=0, \dots, t-1}$ を公開 (VSS)
 - 受け取ったユーザはシェアの正しさを確認する
- ユーザ k はユーザ $j (\neq k)$ からのシェア $f_j(u_k)$ を全て足す
 $s_k := \sum_j f_j(u_k)$ を自身の最終シェアとする
 - 対応する検証鍵 $s_k P_0 = \sum_j f_j(u_k) P_0$ は誰でも計算可能
- $s := \sum_{k=1}^n s_{k,0}$ が全体の秘密鍵, $P := \sum_{k=1}^n s_k P_0 = s P_0$ が公開鍵
- 秘密分散の線型性から $\{s_k\}$ は s のシェアとなっている



後出しの不正を防ぐ

様々な改善

- ハッシュを使って後出しジャンケンを防止
 - $h_i := \text{Hash}(s_{k,0}P_0, \dots, s_{k,t-1}P_0)$ を全員が公開した後, $\{s_{k,i}P_0\}$ を公開して検証
 - より安全にはコミットメント（余裕があれば後の講義で）を使う
- 違反者の除外
- ユーザ k のPoPを実行
- 詳細は [R.Gennaro, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", 2006](#)など