Cryptographic Primitives used in PPBA

Andriy Oliynyk
a.oliynyk@samsung.com

Samsung Research and Development Institute, Ukraine

Contents

Mathematical Introduction

Elliptic curves I

- Equation E of the curve:

$$y^2 = x^3 + ax + b, \quad a, b \in \mathbb{F}_p, \quad 4a^3 + 27b^2 \neq 0.$$

- Points: all pairs $(\alpha, \beta) \in \mathbb{F}_p^2$ such that

$$\beta^2 = \alpha^3 + a\alpha + b,$$

together with zero point $\mathcal{O} = (0, 0)$.

- Notation:

$$E, E/\mathbb{F}_p, E(\mathbb{F}_p).$$

Elliptic curves II

For $\mathcal{P}, \mathcal{Q} \in E$ the addition rule is:

- $\mathcal{P} + \mathcal{O} = \mathcal{O} + \mathcal{P} = \mathcal{P}$;
- if $\mathcal{P} = (\alpha, \beta)$, $\mathcal{Q} = (\alpha, -\beta)$, then $\mathcal{P} + \mathcal{Q} = \mathcal{O}$;
- if $\mathcal{P} = (\alpha_1, \beta_1)$, $\mathcal{Q} = (\alpha_2, \beta_2)$, $\alpha_1 \neq \alpha_2$, then $\mathcal{P} + \mathcal{Q} = \mathcal{R} = (\alpha_3, \beta_3)$, where

$$\alpha_3 = \lambda^2 - \alpha_1 - \alpha_2, \quad \beta_3 = \lambda(\alpha_1 - \alpha_3) - \beta_1, \quad \lambda = \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1};$$

- if $\mathcal{P} = \mathcal{Q} = (\alpha_1, \beta_1)$ i $\beta_1 \neq 0$, then $\mathcal{P} + \mathcal{Q} = \mathcal{R} = (\alpha_3, \beta_3)$, where

$$\alpha_3 = \mu^2 - 2\alpha_1, \quad \beta_3 = \mu(\alpha_1 - \alpha_3) - \beta_1, \quad \mu = \frac{3\alpha_1^2 + a}{2\beta_1}.$$

Elliptic curves III

Group of points

The set E with respect to defined addition of points form an abelian group.

Notation:

$$g^n = \underbrace{g + \ldots + g}_{n \text{ times}}, g \in E, n \in \mathbb{Z}.$$

$E/\mathbb{F}_q$ always contains an element of prime order $\simeq q$.

Bilinear pairing

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ be three cyclic groups of order q for some large prime q. A bilinear pairing is a map $B : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ satisfying the following properties.

- **Bilinear** $B(P^a, Q^b) = B(P, Q)^{ab}$ for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}$.
- **Non-degenerate** The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to the identity in $\mathbb{G}_3$.
- **Computable** There is an efficient algorithm to compute $B(P, Q)$ for any $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$.

Discrete logarithm

For a cyclic group

$$\mathbb{G} = \langle g \rangle$$

the discrete logarithm problem with respect to base g is

For a given $h \in \mathbb{G}$ find $a \in \mathbb{Z}$ such that

$$g^a = h.$$

Notation: $a = DL_g(h)$.

Residues

Let q be a prime.

- The set

$$\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$$

  is called the set of residues modulo q.
- Addition and multiplication are well-defined on $\mathbb{Z}_q$ (*add or multiply as integers and take residues modulo* q).
- $\mathbb{Z}_q$ is a cyclic group with respect to addition.
- $\mathbb{Z}_q$ is a field with respect to addition and multiplication.
- The set of vectors of length n over $\mathbb{Z}_q$ is

$$\mathbb{Z}_q^n = \{(x_1, \ldots, x_n) : x_1, \ldots, x_n \in \mathbb{Z}_n\}.$$

  Notation:

$$\bar{x} = (x_1, \ldots, x_n).$$

Cryptographic Introduction

Syntax of Secret-key Cryptography

Space of keys $\mathcal{K}$, space of plaintexts $\mathcal{M}$, space of ciphertexts $\mathcal{C}$.
Three algorithms **Setup**, **Enc**, **Dec**.

- Initialization algorithm **Keygen** outputs public parameters and a secret key sk $\in \mathcal{K}$.
- Encryption algorithm **Enc** takes the key sk and a plaintext $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$.
- Decryption algorithm **Dec** takes the key sk and a ciphertext $c \in \mathcal{C}$ and outputs a plaintext $m \in \mathcal{M}$.

Correctness requirement

$$\mathbf{Dec}_{\mathsf{sk}}(\mathbf{Enc}_{\mathsf{sk}}(m)) = m.$$

Kerckhoffs' principle: all algorithms **Keygen**, **Enc**, **Dec** are public and the only secret is the key sk.

Syntax of Public-key Cryptography

Spaces of secret keys $\mathcal{SK}$ and public keys $\mathcal{PK}$, space of plaintexts $\mathcal{M}$, space of ciphertexts $\mathcal{C}$.
Again three (probabilistic) algorithms **Keygen**, **Enc**, **Dec**.

- Initialization algorithm **Keygen** outputs public parameters and a secret key $sk \in \mathcal{SK}$ and a public key $pk \in \mathcal{PK}$.
- Encryption algorithm **Enc** takes the public key pk and a plaintext $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$.
- Decryption algorithm **Dec** takes the secret key sk and a ciphertext $c \in \mathcal{C}$ and outputs a plaintext $m \in \mathcal{M}$.

Correctness requirement

$$\Pr[\mathbf{Dec}_{sk}(\mathbf{Enc}_{pk}(m)) = m] = 1.$$

• All algorithms **Keygen**, **Enc**, **Dec** and key pk are public and the only secret is the secret key sk.

Syntax of Inner-Product Cryptography

Spaces of master secret keys $\mathcal{MSK}$, decryption keys $\mathcal{DK}$ and space of plaintexts $\mathbb{Z}_q^n$.

This time four (probabilistic) algorithms **Setup**, **Keygen**, **Enc**, **Dec**.

- Initialization algorithm **Setup** outputs public parameters and a master secret key *msk*.
- Key generation algorithm **Keygen** takes the master secret key *msk* and a vector $\bar{y} \in \mathbb{Z}_q^n$ and outputs a decryption key $dk \in \mathcal{DK}$.
- Encryption algorithm **Enc** takes the master secret key *msk* and a plaintext $\bar{x} \in \mathbb{Z}_q^n$ and outputs a ciphertext $c \in \mathcal{C}$.
- Decryption algorithm **Dec** takes a decryption key *dk* and a ciphertext $c \in \mathcal{C}$ and outputs a value $z \in \mathbb{Z}_q$.

Correctness requirement

$$\Pr[\mathbf{Dec}_{dk}(\mathbf{Enc}_{msk}(\bar{x})) = \langle \bar{x}, \bar{y} \rangle] = 1,$$

where

$$\langle \bar{x}, \bar{y} \rangle = x_1 y_1 + \ldots + x_n y_n.$$

Composable Inner-Product Encryption (CIPE)

Public Parameters I

- p — prime number, the cardinality of the Galois field $\mathbb{F}_p$, in current implementation

$$p = 36u^4 + 36u^3 + 24u^2 + 6u + 1,$$

where

$$u = -(2^{62} + 2^{55} + 1),$$

in decimal expansion

$$p = 16798108731015832284940804142231733909889187121439069848933715426072753864723$$

Public Parameters II

- $\mathbb{F}_{p^{12}}$ — the Galois field, constructed as the following tower of extensions

$$\mathbb{F}_{p^2} = \mathbb{F}_p[x]/(x^2+1),$$
$$\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[y]/(y^3-x-1),$$
$$\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[z]/(z^2-y);$$

Public Parameters III

- `CurveFp254BN2` — the elliptic curve over $\mathbb{F}_p$ defined by the equation

$$y^2 = x^3 + 2$$

- q — prime number, the order of the cyclic groups to be considered,

$$q = 36u^4 + 36u^3 + 18u^2 + 6u + 1;$$

- $\mathbb{G}_1$ — the cyclic group of order q of points of the elliptic curve `CurveFp254BN2`

Public Parameters IV

- g, h — generating elements of $\mathbb{G}_1$;
- $\mathbb{G}_2$ — the cyclic group isomorphic to $\mathbb{G}_1$ via the Frobenius authomorphism;
- $\bar{g}, \bar{h}$ — generating elements of $\mathbb{G}_2$;
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}^*_{p^{12}}$ — the Optimal Ate pairing;
- $\mathbb{G}_T$ — the cyclic group isomorphic to $\mathbb{G}_1$ via Optimal Ate bilinear pairing;
- g_T — a generating element of $\mathbb{G}_T$;
- n — dimension of a biometric template.

ElGamal Secret-Key Encryption I

- Space of keys: $\mathbb{Z}_q$
- Space of plaintexts: $\mathbb{Z}_q^n$
- Spaces of ciphertexts: $\mathbb{G}_1^{2n}$, $\mathbb{G}_2^{2n}$
- Key generation (run on Device)

ElG.Keygen

$\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_q$

return sk

ElGamal Secret-Key Encryption II

- Encryption (run on Scanner during registration)

ElG.Enc$(\mathsf{sk}, \bar{y} = (y_1, \ldots, y_n))$

$\bar{r} = (r_1, \ldots, r_n) \leftarrow_\$ \mathbb{Z}_q^n$

$h = g^{\mathsf{sk}}$

For $i = 1$ to n do

$c_i = g^{y_i} \cdot h^{r_i}, c_{n+i} = g^{r_i}$

$\bar{c} = (c_1, \ldots, c_{2n})$

return $\bar{c}$

ElGamal Secret-Key Encryption III

- Encryption (run on Scanner during authentication)

$$\underline{\mathbf{ElG.Enc}(\mathsf{sk}, \bar{x} = (x_1, \ldots, x_n))}$$

$$\bar{r} = (r_1, \ldots, r_n) \leftarrow_{\$} \mathbb{Z}_q^n$$

$$\bar{h} = \bar{g}^{\mathsf{sk}}$$

For $i = 1$ to n do

$$c_i = \bar{g}^{x_i} \cdot \bar{h}^{r_i}, c_{n+i} = \bar{g}^{r_i}$$

$$\bar{c} = (c_1, \ldots, c_{2n})$$

return $\bar{c}$

ElGamal Secret-Key Encryption IV

- Decryption (not used):

ElG.Dec$(\mathsf{sk}, \bar{c} = (c_1, \ldots, c_{2n}))$

For $i = 1$ to n do

$x_i = DL_g(c_i \cdot c_{n+i}^{-\mathsf{sk}})$

$\bar{x} = (x_1, \ldots, x_n)$

return $\bar{x}$

Composable Inner-Product Encryption I

- Space of master keys:

$$\mathbb{G}_1^n \times \mathbb{Z}_q^n \times \mathbb{Z}_q^n \times \mathbb{G}_2^{n+2} \times \mathbb{Z}_q^{n+2} \times \mathbb{Z}_q^{n+2};$$

- Spaces of messages: $\mathbb{G}_1^{2n}$, $\mathbb{G}_2^{2n}$;
- Space of decryption keys: $\mathbb{G}_1^{2(n+4)}$;
- Space of ciphertexts $\mathbb{G}_2^{2(n+4)}$.

Composable Inner-Product Encryption II

- Master key generation (run on Device)

CIPE.Setup

$\bar{s} = (s_1, \ldots, s_n) \leftarrow_{\$} \mathbb{Z}_q^n$

$\bar{s} = (t_1, \ldots, t_n) \leftarrow_{\$} \mathbb{Z}_q^n$

$\bar{u} = (u_1, \ldots, u_{n+2}) \leftarrow_{\$} \mathbb{Z}_q^{n+2}$

$\bar{v} = (v_1, \ldots, v_{n+2}) \leftarrow_{\$} \mathbb{Z}_q^{n+2}$

For $i = 1$ to n do

$h_i = g^{s_i} h^{t_i}$

For $i = 1$ to $n + 2$ do

$\bar{h}_i = \bar{g}^{u_i} \bar{h}^{v_i}$

$\mathrm{msk} = (\{h_i, s_i, t_i\}_{i=1}^{n}, \{\bar{h}_i, u_i, v_i\}_{i=1}^{n+2})$

return msk

Composable Inner-Product Encryption III

- Decryption key generation (run on Device during registration)

CIPE.Conv.DK$(\mathsf{msk}, \bar{c} = (c_1, \ldots, c_{2n}))$

$$(r_0, r_1) \leftarrow_\$ \mathbb{Z}_q^2$$

$$d_{0,1} = g^{r_0}, d_{0,2} = h^{r_0}$$

$$d_{1,1} = g^{r_1}, d_{1,2} = h^{r_1}$$

$$d_{0,3} = \left(\prod_{j=1}^{n} c_j^{-s_j}\right) \cdot h_1^{r_0}, d_{0,4} = \left(\prod_{j=1}^{n} c_j^{-t_j}\right) \cdot h_2^{r_0}$$

$$d_{1,3} = \left(\prod_{j=1}^{n} c_{n+j}^{-s_j}\right) \cdot h_1^{r_1}, d_{1,4} = \left(\prod_{j=1}^{n} c_{n+j}^{-t_j}\right) \cdot h_2^{r_1}$$

For $i = 1$ to n do

$$d_{0,i+4} = c_i \cdot h_{i+2}^{r_0}, d_{1,i+4} = c_{n+i} \cdot h_{i+2}^{r_0}$$

$$DK_0 = (d_{0,1}, \ldots d_{0,n+4}), DK_1 = (d_{1,1}, \ldots d_{1,n+4})$$

return (DK_0, DK_1)

Composable Inner-Product Encryption IV

- Encryption (run on Device during authentication)

CIPE.Conv.Enc(msk, $\bar{c} = (c_1, \ldots, c_{2n})$)

$(r_0, r_1) \leftarrow_{\$} \mathbb{Z}_q^2$

$c_{0,3} = \bar{g}^{r_0}, c_{0,4} = \bar{h}^{r_0}$

$c_{1,3} = \bar{g}^{r_1}, c_{1,4} = \bar{h}^{r_1}$

For $i = 1$ to n do

$c_{0,i+4} = c_i \cdot \bar{h}_i^{r_0}, c_{1,i+4} = c_{n+i} \cdot \bar{h}_i^{r_1}$

$$c_{0,1} = \left(\prod_{j=3}^{n+4} c_{0,j}^{-u_j} \right), c_{0,2} = \left(\prod_{j=3}^{n+4} c_{0,j}^{-v_j} \right)$$

$$c_{1,1} = \left(\prod_{j=3}^{n+4} c_{1,j}^{-u_j} \right), c_{1,2} = \left(\prod_{j=3}^{n+4} c_{1,j}^{-v_j} \right)$$

$CT_0 = (c_{0,1}, \ldots c_{0,n+4}), CT_1 = (c_{1,1}, \ldots c_{1,n+4})$

return (CT_0, CT_1)

Composable Inner-Product Encryption V

- Decryption (run on Server during authentication*)

CIPE.Dec$(\mathsf{sk}, (DK_0, DK_1), (CT_0, CT_1))$

For $i = 1$ to $n + 4$ do

$dk_i = d_{0,i} \cdot d_{1,i}^{-\mathsf{sk}}$(* *run during registration*)

$ct_i = c_{0,i} \cdot c_{1,i}^{-\mathsf{sk}}$

$$d = \left(\prod_{i=1}^{n+2} e(dk_i, ct_i) \right)$$

$g_T = e(g, \bar{g})$

$a = DL_{g_T}(d)$

return a

Thank you!