[LNR - CCS18]

Fast Secure Multiparty ESDSA

발표자 : 한양대학교 수학과 주찬양

Contents

1. Comparing with [GG-CCS18]
2. Main Idea
3. Operations of Functionality $\mathcal{F}_{\text{mult}}$
4. The Protocol for $\mathcal{F}_{\text{ECDSA}}$
5. Experimental Results

1. Comparing with [GG-CCS18]

Assumption

Security

1. Comparing with [GG–CCS18]

	[GG–CCS18]	[LNR–CCS18]
Assumption	◆ DDH ◆ Paillier-EC ◆ Strong RSA	◆ DDH ◆ Indistinguishability of Paillier
Security	◆ Game Base	◆ Simulation Base

2. Main Idea

Let *Gq: group of order q,* x: secret key, h: public key, g: generator, m: message, r: randomness

Multiplicatively Homorphic ElGamal

Define E : (Gq, *)→(Gq x Gq, *)
such that E(m) = (g^r, m * h^r)

Then we can check :

$$(g^{r1}, m1 * h^{r1}) * (g^{r2}, m2 * h^{r2})$$
$$= (g^{r1+r2}, (m1 * m2) * h^{r1+r2})$$

Hence,

E(m1) * E(m2) = E(m1 * m2)

2. Main Idea

Let *Gq: group of order q,* x: secret key, h: public key, g: generator, m: message, r: randomness

Multiplicatively Homorphic ElGamal

Define E : (Gq, *)→(Gq x Gq, *)
such that $E(m) = (g^{r}, m * h^{r})$

Then we can check :

$$(g^{r1}, m1 * h^{r1}) * (g^{r2}, m2 * h^{r2}) = (g^{r1+r2}, (m1 * m2) * h^{r1+r2})$$

Hence,

$$E(m1) * E(m2) = E(m1 * m2)$$

Additively Homomorphic ElGamal

Define E : (Gq, *)→(Gq x Gq, +)
such that $E(m) = (g^{r}, g^{m} * h^{r})$

Then, we can check :

$$(g^{r1}, g^{m1} * h^{r1}) * (g^{r2}, g^{m2} * h^{r2}) = (g^{r1+r2}, g^{m1+m2} * h^{r1+r2})$$

Hence,

$$E(m1)*E(m2) = E(m1 + m2)$$

3. Operations of Functionality $\mathcal{F}_{\text{mult}}$

Init

- Input (G, g, q)
- Private share x_i s.t. $\sum x_i = x$

Input

- Input (input, sid, s_i)
- Compute $EG_{enc}(s_i ;r_i)$, $s_{sid} = \sum s_i$

Affine

- Input (sid1, sid2, x, y)
- $s_{sid2} = s_{sid1} \cdot x + y$ *mod q*

Mult

- Input (mult, sid1, sid2)
- In Parallel

  <Compute Product of shares>

  - Make share c_i s.t. $\sum c_i = s_{sid1} \cdot s_{sid2}$ *mod q*
  - Send c to all parties

  <Compute ElGamal in the Exponent>

  - Get $EG_{enc}(a*b)$ using add. property
- Verity correctness

Element-out

- Input (sid)
- $P = s_{sid} \cdot G$

3. Operations of Functionality $\mathcal{F}_{\text{mult}}$

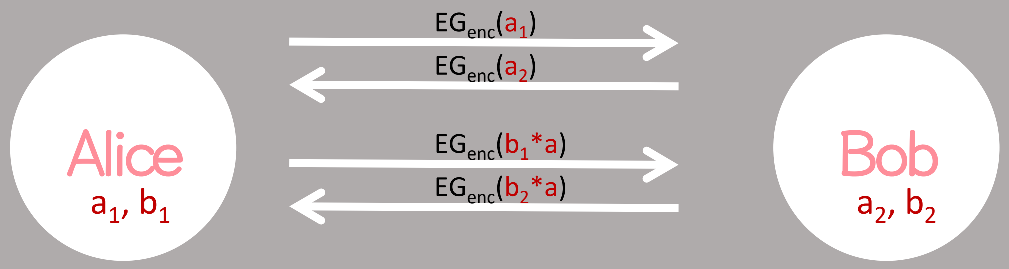
Goal : Want to get correct $c = a\cdot b$ while unrevealing a , b

⟨Private Multiplication to Get Product of Shares⟩

Computes

$\pi_{\text{mult}}((a_1, a_2),(b_1, b_2)) = (c_1, c_2)$ s.t. $c = \sum c_i$

⟨Computing ElGamal in the exponent⟩



Alice

a_1, b_1

$EG_{enc}(a_1)$

$EG_{enc}(a_2)$

$EG_{enc}(b_1*a)$

$EG_{enc}(b_2*a)$

Bob

a_2, b_2

$EG_{enc}(a_1)*EG_{enc}(a_2) = EG_{enc}(a_1+a_2)$

$b_i*EG_{enc}(a) = EG_{enc}(b_i*a)$

$EG_{enc}(b_1*a)*EG_{enc}(b_2*a) = EG_{enc}((b_1+b_2)*a)$

Eventually obtains $EG_{enc}(a*b)$

Get (c_1, c_2) s.t. $a\cdot b = c_1 + c_2$

Get $ab\cdot G$

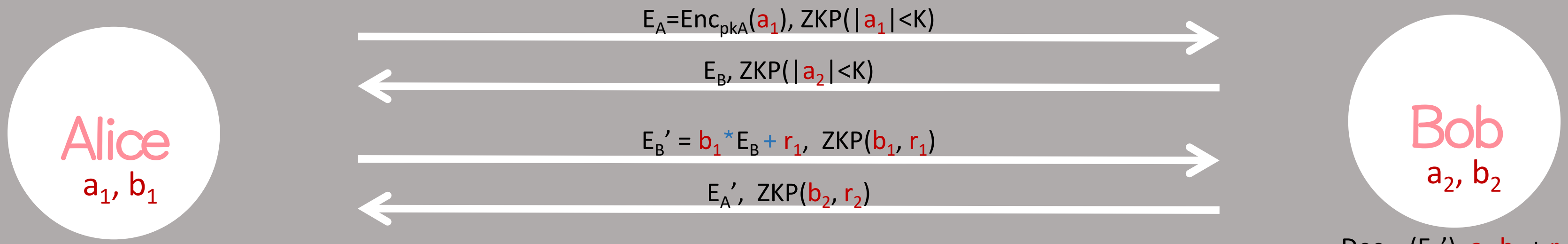
Check : $c\cdot G = ab\cdot G$

3. Operations of Functionality $\mathcal{F}_{\text{mult}}$

⟨Instantiation of Private Multiplication π_{mult}⟩

- ➡ Paillier
- – Oblivious Transfer

Computes $\pi_{\text{mult}}((a_1, a_2),(b_1, b_2)) = (c_1, c_2)$ s.t. $c = \sum c_i$



Alice a_1, b_1

Bob a_2, b_2

$E_A = Enc_{pkA}(a_1)$, $ZKP(|a_1|<K)$

E_B, $ZKP(|a_2|<K)$

$E_B' = b_1 * E_B + r_1$, $ZKP(b_1, r_1)$

E_A', $ZKP(b_2, r_2)$

$Dec_{skA}(E_A') = a_1 \cdot b_2 + r_2$

$Dec_{skB}(E_B') = a_2 \cdot b_1 + r_1$

Let $c_1 = Dec_{skA}(E_A') + a_1 \cdot b_1 - r_1$
$= a_1 \cdot b + r_2 - r_1$

Let $c_2 = a_2 \cdot b - r_2$

Then we have $c_1 + c_2 = a \cdot b$

3. Operations of Functionality $\mathcal{F}_{\text{mult}}$

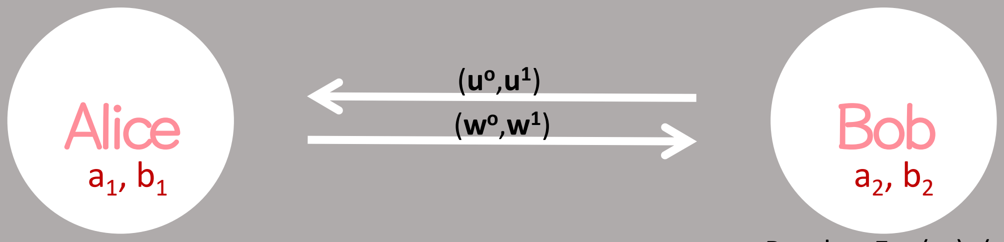
⟨Instantiation of Private Multiplication π_{mult}⟩

- Paillier
- ➡ Oblivious Transfer

Computes $\pi_{\text{mult}}((a_1, a_2),(b_1, b_2)) = (c_1, c_2)$ s.t. $c = \sum c_i$

RandomEnc

- Input $x \in \mathbb{Z}_q$
- Choose a random $\mathbf{r} \leftarrow \{0,1\}^n$
- Select a vector $\mathbf{a} \in \mathbb{Z}_q^n$ s.t. $x = \sum a_i$
- Select a random vector pair $(\mathbf{a^0},\mathbf{a^1})$ s.t. $a_i^{r_i} = a_i$
- Output $(\mathbf{a^0},\mathbf{a^1},\mathbf{r})$



$\mathbf{t} \leftarrow \mathbb{Z}_q^n$

Compute $\mathbf{w^0} = \mathbf{u^0} \cdot b_1 + \mathbf{t}$

$\mathbf{w^1} = \mathbf{u^1} \cdot b_1 + \mathbf{t}$

Let $y_A = -\sum t_i$

RandomEnc(a_2)=$(\mathbf{u^0},\mathbf{u^1},\mathbf{r})$

Let $y_B = \sum w_i^{r_i}$

$y_A + y_B = a_2 \cdot b_1$

Similary, $z_A + z_B = a_1 \cdot b_2$

Then $c_1 = a_1 \cdot b_1 + y_A + z_A$

$c_2 = a_2 \cdot b_2 + y_B + z_B$

4. The Protocol for $\mathcal{F}_{\text{ECDSA}}$

Init
- Input (G, g, q)
- Private share x_i s.t. $\sum x_i = x \cdot G$

Input
- Input (input, sid, s_i)
- Compute $EG_{enc}(s_i ; r_i)$, $s_{sid} = \sum s_i$

Affine
- Input (sid1, sid2, x, y)
- $s_{sid2} = s_{sid1} \cdot x + y \ \mathit{mod}\ q$

Mult
- Input (mult, sid1, sid2)
- In Parallel

  <Compute Product of shares>

  - Make share c_i s.t. $\sum c_i = s_{sid1} \cdot s_{sid2}$

  - Send c to all parties

  <Compute ElGamal in the Exponent>

  - Get $EG_{enc}(a * b)$ using add. property
- Verity correctness

El-out
- Input (sid)
- $P = s_{sid} \cdot G$

KETGEN → $sk_i = x_i$, $pk = Q$ s.t. $Q = G \cdot \sum x_i$

F_{mult}(Init) ⟹ Private share x_i ($sk = \sum x_i$, $pk = x \cdot G$)

F_{mult}(E-out) ⟹ $Q = x \cdot G$

Output Q

4. The Protocol for $\mathcal{F}_{\text{ECDSA}}$

Init
- Input (G, g, q)
- Private share x_i s.t. $\sum x_i = x \cdot G$

Input
- Input (input, sid, s_i)
- Compute $EG_{enc}(s_i ;r_i)$, $s_{sid} = \sum s_i$

Affine
- Input (sid1, sid2, x, y)
- $s_{sid2} = s_{sid1} \cdot x + y$ *mod q*

Mult
- Input (mult, sid1, sid2)
- In Parallel

  <Compute Product of shares>

  - Make share c_i s.t. $\sum c_i = s_{sid1} \cdot s_{sid2}$
  - Send c to all parties

  <Compute ElGamal in the Exponent>

  - Get $EG_{enc}(a*b)$ using add. property
- Verity correctness

El-out
- Input (sid)
- $P = s_{sid} \cdot G$

SIGN → $(r, k^{-1} \cdot \rho^{-1} \cdot \rho \cdot (H(m) + r \cdot x))$

F_{mult}(Input) ⇒ Random k, ρ

F_{mult}(E-out) ⇒ $R = k \cdot G$

Let $R = (r_x, r_y)$ and $r = r_x \mod q$

F_{mult}(Affine) ⇒ $H(m) + r \cdot x$

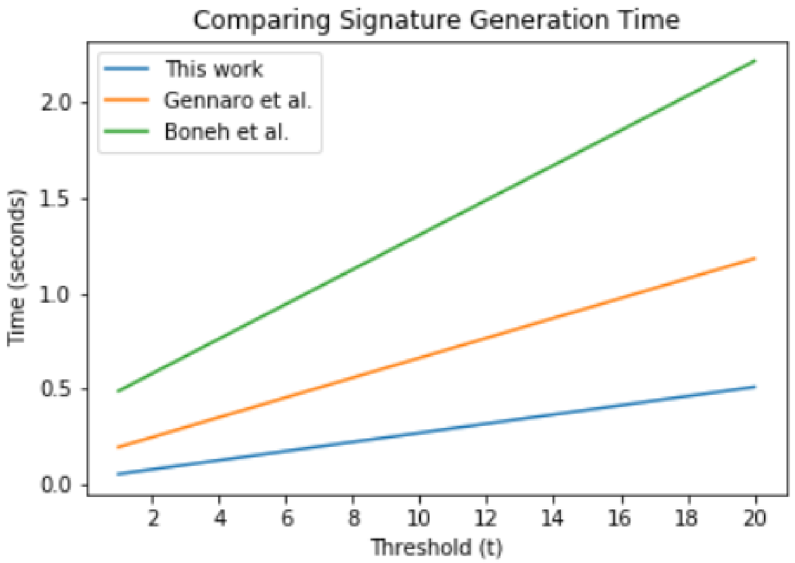
F_{mult}(Mult) ⇒ $\tau = k \cdot \rho$

Compute τ^{-1}

F_{mult}(Mult) ⇒ $\beta = \rho \cdot (H(m) + r \cdot x)$

Output $(r, \tau^{-1} \cdot \beta)$

5. Experimental Results



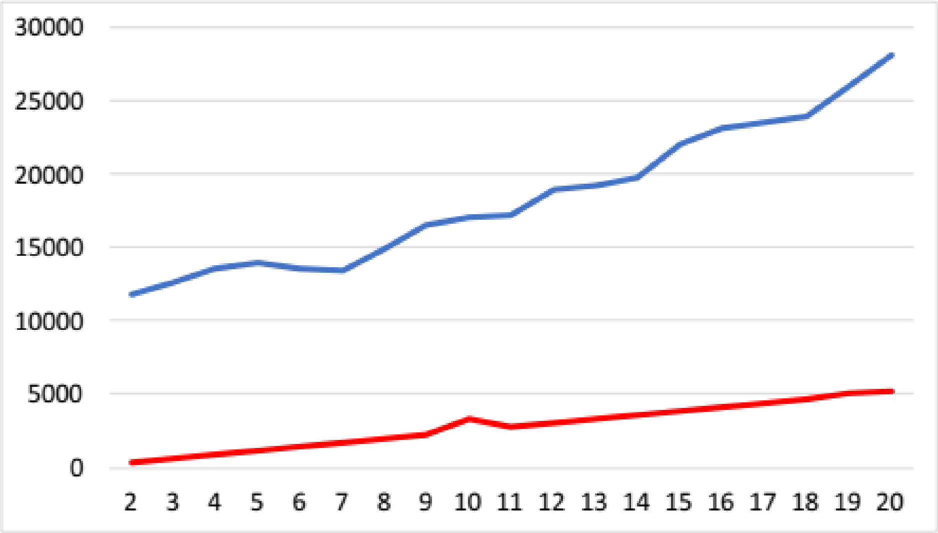


Figure 1: The running times in milliseconds for key generation (top line in blue) and signing (bottom line in red) for 2-20 parties, for the Pailler variant of the protocol.

[GG – CCS18]

[LNR – CCS18]

Q&A