Practical Two-level Homomorphic Encryption in Prime-order Bilinear Groups

Goichiro Hanaoka[*1]

Joint-work-with: Nuttapong Attrapadung[*1], Shigeo Mitsunari[*2], Yusuke Sakai[*1], Tadanori Teruya[*1]

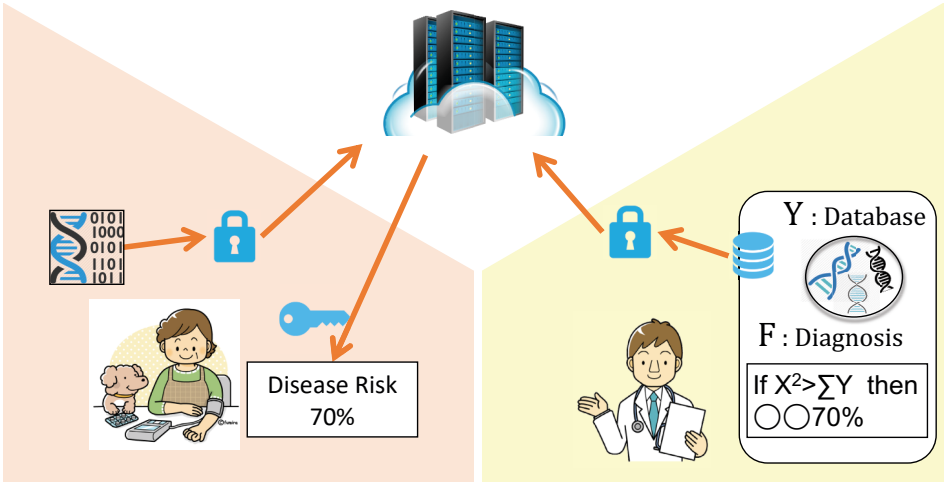
*1 AIST, *2 Cybozu labs

Outline

- Background
- Two-level Homomorphic encryption
- An efficient construction
- Security
- Implementation
- Conclusion

Background

Computing on encrypted data



- Data analysis with taking care of sensitive data

Homomorphic Encryption (HE)

- Allows computation on encrypted data
- Many applications related to privacy-preserving schemes
- Types of HE
  - Additively HE (ex. Goldwasser-Micali, Okamoto-Uchiyama, Paillier, Lifted-ElGamal)
    - $\mathrm{Enc}(m) + \mathrm{Enc}(m') = \mathrm{Enc}(m + m')$
  - Multiplicatively HE (ex. RSA, ElGamal)
    - $\mathrm{Enc}(m) \times \mathrm{Enc}(m') = \mathrm{Enc}(mm')$
  - Fully HE (ex. Gentry, BGV, BV, GSW, …)
    - Can do homomorphic add. and mult.

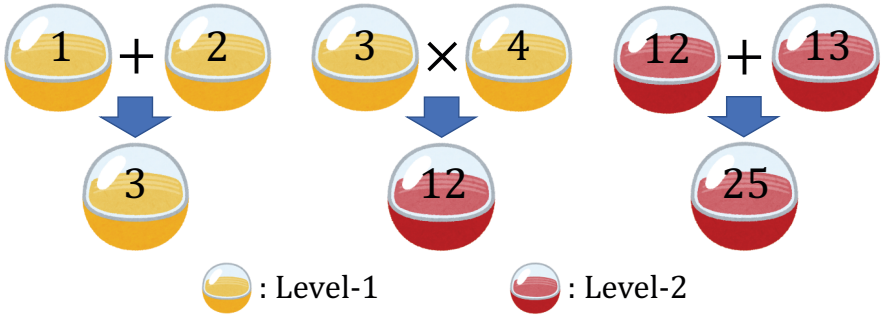
Pros and Cons

- Add. HE, Mult. HE
  - Applications are restricted
- Fully HE (FHE)
  - Any computations possible, but inefficient
  - Security relies on less standard assumptions
- Leveled HE
  - The number of homomorphic mult. is restricted.
  - An intermediate notion between A/M HE and FHE.

	A/M HE	Leveled HE	FHE
Efficiency	very good	medium	bad
Functionality	medium	good	very good

Two-level HE

- HE that allows *one* homomorphic multiplication



- Allows degree-2 polynomial homomorphic evaluations
- Allows inner product of two vectors
- $x = (x_1, x_2, \dots), y = (y_1, y_2, \dots)$
- $\sum_i \mathrm{Enc}_1(x_i) \times \mathrm{Enc}_1(y_i) = \mathrm{Enc}_2(\sum_i x_i \times y_i)$

Applications

- Secure 2-DNF formula evaluation
- Delegated secure inner-product on encrypted data
- Efficient (symmetric) private information retrieval
- Cross tabulation on encrypted data
- Efficient election protocol
- …

Existing Two-level HE

- Boneh, Goh, Nissim (TCC 2005)
  - Based on Composite-order pairings, hence much less efficient
- Freeman (EUROCRYPT 2010)
  - Composite-to-prime-order transformation framework, applied to BGN
- Herold, Hesse, Hofheinz, Rafols, Rupp (CRYPTO 2014)
  - Improving Freeman's frameworks
  - Only Type 1 pairings, inefficient
- Catalano, Fiore (ACM CCS 2015)
  - Transformation from d-Level HE to (2d)-level
  - Instantiations are not necessarily efficient
- AHM+ (AsiaCCS 2018): This talk
  - Efficient construction based on the lifted-ElGamal encryption
  - Portable high-speed implementations
- Note:
  - Decryption in all these schemes requires discrete log (DL)
  - Hence plaintext space should be sufficiently small (up to 32-bit)

An Efficient Construction of Two-level HE

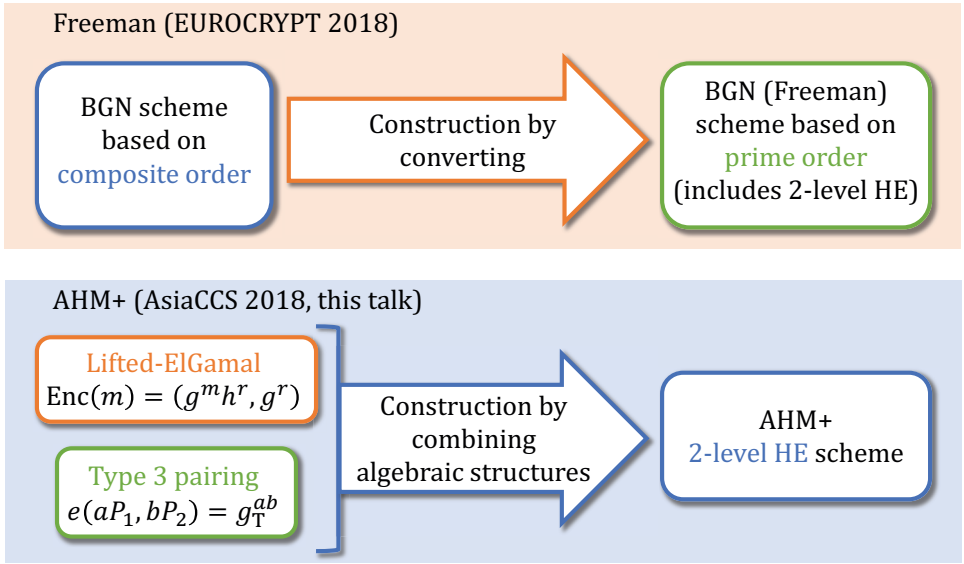
Basic Idea

- Existing schemes
  - Establish a "broader fundamental & theoretical framework"
  - Then, construct L2HE as an "application"
- Our scheme
  - Concentrate on "L2HE-dedicated design"
  - Start from "promising tools" for fast HE, i.e. Type-3 pairing and ElGamal
    - Not general but fully tuned for L2HE

An Efficient Construction

- Combine the lifted-ElGamal encryption scheme with Type 3 pairings
  - First, straightforwardly construct two-level HE
  - Then, consider "simpler" construction
    - While Freeman considered a conversion of composite-to-prime order
  - Level-1 (L1) ciphertext (CT) is same as lifted-ElGamal
  - Format of level-2 (L2) CT is same as Freeman's scheme
- Note: Type 3 pairings
  - Cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order prime p with bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
  - $e(aP, bQ) = e(P, Q)^{ab}$ for $a, b \in \mathbb{Z}_p, P \in \mathbb{G}_1, Q \in \mathbb{G}_2$
  - $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficient map between $\mathbb{G}_1$ and $\mathbb{G}_2$

Summary of Constructions



Setup and Key Generation

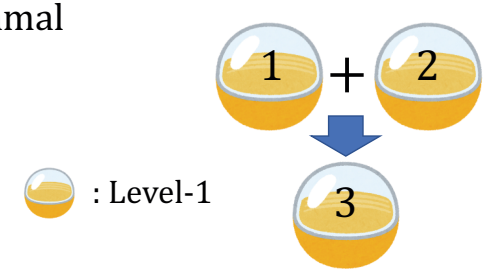
- Setup
  - Cyclic group $\mathbb{G}_i = \langle P_i \rangle$ over an elliptic curve with prime order p for $i = 1, 2$
  - $\mathbb{G}_T = \langle g_T \rangle$, where $g_T = e(P_1, P_2)$
- Key generation
  - Secret key $s_1, s_2 \in \mathbb{Z}_p$ is generated at random
  - Public key $Q_1 = s_1 P_1, Q_2 = s_2 P_2$ (with optional precomputation $z_1 = g_T, z_2 = g_T^{s_1}, z_3 = g_T^{s_2}, z_4 = g_T^{s_1 s_2}$)
- Note: Colors
  - Green: Public part
  - Blue: Secret and hidden part

Level-1 CT and Enc./Dec.

- Encrypt
  - Plaintext m and randomness r
  - $\mathrm{Enc}_{\mathbb{G}_i}(m) = (mP_i + rQ_i, rP_i)$ for $i = 1, 2$
  - Duplicated form:
    $$\mathrm{Enc}_1(m) := \left(\mathrm{Enc}_{\mathbb{G}_1}(m), \mathrm{Enc}_{\mathbb{G}_2}(m)\right)$$
  - Note: $\mathbb{G}_1$ can be mult. with $\mathbb{G}_2$ only, vice versa, so that duplicated form is needed for general usage
- Decrypt
  - For $i = 1, 2$, decrypt $\mathrm{Enc}_{\mathbb{G}_i}(m) = (S, T)$ by
    $S - s_i T = (mP_i + rQ_i) - s_i(rP_i) = mP_i$
    and then, to obtain m, solve DL
- Almost same as lifted-ElGamal

Homomorphic Addition on L1 CT

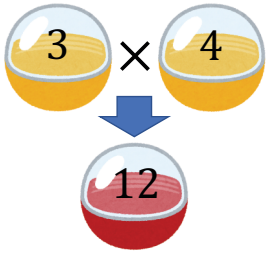
- For $i = 1, 2$,
  $$\begin{aligned}&\mathrm{Enc}_{\mathbb{G}_i}(m_1) + \mathrm{Enc}_{\mathbb{G}_i}(m_2)\\ &= (m_1 P_i + r_1 Q_i, r_1 P_i) + (m_2 P_i + r_2 Q_i, r_2 P_i)\\ &= \left((m_1 + m_2)P_i + (r_1 + r_2)Q_i, (r_1 + r_2)P_i\right)\\ &= \mathrm{Enc}_{\mathbb{G}_i}(m_1 + m_2)\end{aligned}$$
- Also, same as lifted-ElGamal



Homomorphic Multiplication

- $C_1 = (S_1, T_1) = (m_1 P_1 + r_1 Q_1, r_1 P_1) = \mathrm{Enc}_{\mathbb{G}_1}(m_1) \in \mathbb{G}_1^2$
- $C_2 = (S_2, T_2) = (m_2 P_2 + r_2 Q_2, r_2 P_2) = \mathrm{Enc}_{\mathbb{G}_2}(m_2) \in \mathbb{G}_2^2$
- $C_1 \times C_2 := \big(e(S_1, S_2), e(S_1, T_2), e(T_1, S_2), e(T_1, T_2)\big)$
  $= ({z_1}^{m_1 m_2} {z_4}^{\tau'}, {z_2}^{\sigma'}, {z_3}^{\rho'}, {z_1}^{\sigma'+\rho'-\tau'})$
  $= \mathrm{Enc}_2(m_1 m_2) \in \mathbb{G}_T^4$

- $z_1 = g_T, z_2 = {g_T}^{s_1}, z_3 = {g_T}^{s_2}, z_4 = {g_T}^{s_1 s_2}$
- Tensor product of C_1, C_2
- Its result is an level-2 ciphertext



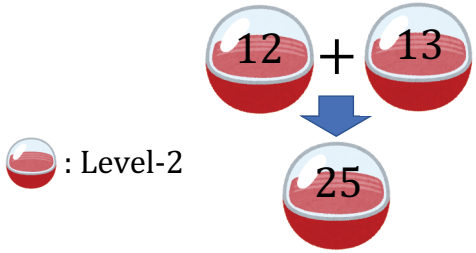
: Level-1

: Level-2

Homomorphic Addition on L2 CT

- $\mathrm{Enc}_2(m_1) + \mathrm{Enc}_2(m_2)$
  $= ({z_1}^{m_1} {z_4}^{\tau_1}, {z_2}^{\sigma_1}, {z_3}^{\rho_1}, {z_1}^{\sigma_1+\rho_1-\tau_1})$
  $+ ({z_1}^{m_2} {z_4}^{\tau_2}, {z_2}^{\sigma_2}, {z_3}^{\rho_2}, {z_1}^{\sigma_2+\rho_2-\tau_2})$
  $= ({z_1}^{m_1+m_2} {z_4}^{\tau_1+\tau_2}, {z_2}^{\sigma_1+\sigma_2},$
  $\quad {z_3}^{\rho_1+\rho_2}, {z_1}^{(\sigma_1+\sigma_2)+(\rho_1+\rho_2)-(\tau_1+\tau_2)})$
  $= \mathrm{Enc}_2(m_1 + m_2)$

- Usual vector addition



: Level-2

Decryption for Level-2 CT

- Decrypting an level-2 ciphertext (c_1, c_2, c_3, c_4)

$$\mathrm{Dec}_2(c_1, c_2, c_3, c_4) := \frac{c_1 {c_4}^{s_1 s_2}}{{c_2}^{s_2} {c_3}^{s_1}}$$
$$= \frac{e(S_1, S_2) e(s_1 T_1, s_2 T_2)}{e(S_1, s_2 T_2) e(s_1 T_1, S_2)}$$
$$= e(S_1 - s_1 T_1, S_2 - s_2 T_2)$$
$$= e(mP_1, m' P_2) = e(P_1, P_2)^{mm'}$$

then solve DLP to obtain mm'

- Note: $(c_1, c_2, c_3, c_4) =$
  $({z_1}^{mm'} {z_4}^{\tau}, {z_2}^{\sigma}, {z_3}^{\rho}, {z_1}^{\sigma+\rho-\tau}) \in \mathbb{G}_T^4,$
  where $z_1 = g_T, z_2 = {g_T}^{s_1}, z_3 = {g_T}^{s_2}, z_4 = {g_T}^{s_1 s_2}$

Size and Benchmark on BN462

	Bit size
Secret key	924
Public key	27720
Dup. L1 CT	5544
L2 CT	22176

	Calc. time in msec
Enc_1	0.452
Enc_2	1.14
Dec_1	9.01
Dec_2	10.01
ReRand_1	0.447
ReRand_2	1.14
Add_1	0.0109
Add_2	0.0231
Mult	8.47

- Note:
  - Use x64 Linux on Core i7-6700
  - Without compressed form
  - Use lookup tables for decryption (20-bit plaintext)

Comparison of Size

SIZE*		
Public key	71%	of Fre10
Secret key	25%	of Fre10
Ciphertext	same	as Fre10

- Fre10: Freemen's scheme (EUROCRYPT 2010)
- Compare bit size on a 462-bit Barreto-Naehrig (BN) curve

Comparison of Time

TIME†		
Key generation	628%	faster than Fre10
Encryption	103%	faster than Fre10
Hom. addition on level-1 CT	same	as Fre10
Hom. addition on level-2 CT	1806%	faster than Fre10
Hom. multiplication	274%	faster than Fre10
Decryption on level-1 CT‡	400%	faster than Fre10
Decryption on level-2 CT‡	533%	faster than Fre10

- CT: Ciphertext
- Fre10: Freemen's scheme in EUROCRYPT 2010
- Compare calculation time on a 462-bit BN curve

Proving the Knowledge of Plaintexts

- Zero-knowledge proof protocols can be applied
- Example 1: Duplicated form of L1 CT
  - Dup. L1 CT is $\left(\mathrm{Enc}_{\mathbb{G}_1}(m), \mathrm{Enc}_{\mathbb{G}_2}(m')\right)$
  - Attach a proof of "$m = m'$"
- Example 2: Proving a CT encrypts a bit
  - Attach a proof of "encrypted plaintext is 0 or 1"
  - Applications: Voting, two-party computation

Proof of Equality

- Duplicated L1 CT:
  - $\left(\mathrm{Enc}_{\mathbb{G}_1}(m), \mathrm{Enc}_{\mathbb{G}_2}(m')\right) = \left((C_1, C_2), (C_3, C_4)\right)$
    $= ((mP_1 + \rho Q_1, \rho P_1), (m'P_2 + \sigma Q_2, \sigma P_2))$
    where $\rho, \sigma \leftarrow \mathbb{Z}_p$ are randomly chosen
  - Should be "$m = m'$"
- Equality can be proved in the same way of NIZK DH-tuple proof

NIZK Proof of Equality

- L1 CT: $\big((C_1, C_2), (C_3, C_4)\big)$
  $= ((mP_1 + \rho Q_1, \rho P_1), (m'P_2 + \sigma Q_2, \sigma P_2))$
- Prove:
  - Randomly choose: $r_\rho, r_\sigma, r_m \leftarrow \mathbb{Z}_p$
  - $(R_1, R_2, R_3, R_4) \leftarrow \big(r_m P_1 + r_\rho Q_1, r_\rho P_1, r_m P_2 + r_\sigma Q_2, r_\sigma P_2\big)$
  - $c \leftarrow H(\text{public param}, C_1, C_2, C_3, C_4, R_1, R_2, R_3, R_4)$
  - $\big(s_\rho, s_\sigma, s_m\big) \leftarrow \big(r_\rho + c\rho, r_\sigma + c\sigma, r_m + cm\big)$
  - Proof $\pi = \big(c, s_\rho, s_\sigma, s_m\big)$
- Verify:
  - $c = H(\text{public param}, C_1, C_2, C_3, C_4, R_1', R_2', R_3', R_4')$
    where
    (R_1', R_2', R_3', R_4')
    $\leftarrow \big(s_m P_1 + s_\rho Q_1 - cC_1, s_\rho P_1 - cC_2, s_m P_2 + s_\sigma Q_2 - cC_3, s_\sigma P_2 - cC_4\big)$

Security

Confidentiality

- Shown scheme is IND-CPA secure under the SXDH assumption
- Note1: IND-CPA (INDistinguishability against Chosen Plaintext Attack)
  - Hidden plaintext from ciphertext
  - Standard base-line security notion
- Note2: SXDH (Symmetric eXternal Diffie-Hellman) assumption
  - $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$, for random α, β, γ,
    $(P_1, \alpha P_1, \beta P_1, \alpha\beta P_1) \approx (P_1, \alpha P_1, \beta P_1, \gamma P_1)$ and
    $(P_2, \alpha P_2, \beta P_2, \alpha\beta P_2) \approx (P_2, \alpha P_2, \beta P_2, \gamma P_2)$
    are computationally indistinguishable

Circuit Privacy

- Shown scheme is circuit private
  - Namely, $\mathrm{ReRand}_i(c) \approx \mathrm{Enc}_i(\mathrm{Dec}_i(c))$
  - Rerandomization: $\mathrm{ReRand}_i(c) \coloneqq c + \mathrm{Enc}_i(0)$
  - $\mathrm{ReRand}_i(c)$ removes a trace of circuit from c
- Note: Arithmetic circuit depends on secret
  - E.g., for $i = 1, 2$, and for a secret integer n,
    $n \times \mathrm{Enc}_i(m) = \sum_{j=1}^{n} \mathrm{Enc}_i(m) = \mathrm{Enc}_i(nm)$
  - Should be $\mathrm{Enc}_i(m) + \mathrm{Enc}_i(m') \approx \mathrm{Enc}_i(m + m')$ and
    $\mathrm{Enc}_1(m) \times \mathrm{Enc}_1(m') \approx \mathrm{Enc}_2(mm')$
  - Note: It is obvious that CTs are in which group
    $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$

Implementation

Practical Two-level Homomorphic Encryption in Prime-order Bilinear Groups

Goichiro Hanaoka[*1]

Joint-work-with: Nuttapong Attrapadung[*1], Shigeo Mitsunari[*2], Yusuke Sakai[*1], Tadanori Teruya[*1]

*1 AIST, *2 Cybozu labs

Our Implementation

- Available in "mcl": A library for pairings
  - C++: https://github.com/herumi/mcl
  - Web browser/Node.js: https://github.com/herumi/she-wasm
  - High-performance implementation for x64/ARM64
  - WebAssembly (wasm)

    - Runs on Microsoft Edge, Firefox, Chrome, Safari *without any plug-ins*
  - Open source: BSD 3-clause

Benchmarks on wasm

- Calculation times in msec
  - Use BN254
  - Use lookup tables for decryption (20-bit plaintext)

	Native (x64)	JavaScript with wasm	
	x64 Linux on Core i7-7700	Firefox on Core i7-7700	Safari on iPhone 7
$\mathrm{Enc}_{\mathbb{G}_1}$	0.018	0.3	0.96
$\mathrm{Enc}_{\mathbb{G}_2}$	0.048	0.82	1.72
$\mathrm{Add}_{\mathbb{G}_1}$	0.00062	0.016	0.016
$\mathrm{Add}_{\mathbb{G}_2}$	0.002	0.036	0.048
Mult	1.17	15.6	24.3
Dec_2	0.66	7.8	12.6

Demo

Oblivious Transfer demo by L2-HE

Input the pos-th digit of π (0 <= pos < 1000000)

pos : 0

query

The result is 0.

status :ok

N = 1000000

M = 0

enc:0msec

Importance of WebAssembly (wasm) Implementation

- Large deployment advantages
  - wasm is a portable and fast binary instruction format
  - Runs on many modern browser
    - Microsoft Edge, Safari, Google Chrome, and Mozilla Firefox on Windows, Linux, macOS, iPhone, Android, and so on…
  - Requires no plugins
  - Being developed as a web standard via the W3C
  - Distribution is easy

Demonstrations of wasm

- Inner product: https://herumi.github.io/she-wasm/she-demo.html
- Oblivious transfer: https://ppdm.jp/ot/

Conclusion

- Practical efficient two-level homomorphic encryption
  - Many times add. and *one-time* mult. on encrypted data
  - Based on Type 3 (asymmetric) pairing
  - Combine the lifted-ElGamal encryption scheme
  - Faster than Freeman's scheme (EUROCRYPT 2010)
- Portable high-performance implementation
  - C++/asm/WebAssembly
  - https://github.com/herumi/mcl
  - https://github.com/herumi/she-wasm
  - Open source: BSD 3-clause

Thank you!