# (Threshold-)CL: (Threshold) Linearly Homomorphic PKE from Class Groups

**CL-Team**

Cyril Bouvier[1], Lennart Braun[3], Guilhem Castagnos[2], **Quentin Combal**[1], Fabien Laguillaumie[1], Kelsey Melissaris[3], Ida Tucker

[1]**Université de Montpellier, CNRS, LIRMM @ Montpellier, France**

[2]Université de Bordeaux, INRIA, CNRS, IMB UMR 5251, F-33405 @ Talence, France

[3]Université Paris Cité, CNRS, IRIF @ Paris, France

2026-10-07 @ NIST Threshold Call Preview Talks #3

# Threshold Call Proposal

- Two "gadget" crypto-systems - category S7
- Castagnos-Laguillaumie (CL) encryption
  - Linearly Homomorphic Encryption scheme
  - Based on Class Groups of imaginary quadratic fields
- Threshold variant: Threshold-CL
  - Threshold LHE
  - Single-round decryption
  - Honest majority ($t < n/2$)
  - Static security (adaptive is in the works)

# Outline

Threshold Linear Homomorphic Encryption

CL

Threshold CL

# Linearly Homomorphic Encryption

## Linearly Homomorphic Encryption (LHE)

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, publish $pk$
- Linear operations on encrypted data

$$\mathsf{Enc}_{pk}(m_1) \otimes \mathsf{Enc}_{pk}(m_2) \rightsquigarrow \mathsf{Enc}_{pk}(m_1 + m_2)$$

$$\mathsf{Enc}_{pk}(m)^{\otimes a} \rightsquigarrow \mathsf{Enc}_{pk}(a \cdot m)$$

## Threshold LHE

- Distributed Key Generation (DKG): $n$ parties $\rightsquigarrow$ shared sk
- Distributed Decryption with threshold $t$
  - $\leq t$ parties $\rightsquigarrow \emptyset$ info on $m$
  - $t + 1$ parties $\rightsquigarrow m$

# LHE applications

- Building MPC protocols for general functions Cramer, Damgård, Nielsen EC'01
- Electronic voting: Helios or Belenios
- Secure Machine Learning
  - "SecureML: A System for Scalable Privacy-Preserving Machine Learning" Mohassel and Zhang, IEEE S&P '17,
- Thresholdized ECDSA
  - From Paillier Gennaro, Goldfeder, ACM CCS'18
  - From CL Castagnos, Catalano, Laguillaumie, Savasta, Tucker, PKC'20

# Outline

Threshold Linear Homomorphic Encryption

CL

Threshold CL

# Class Groups: The Gist

Class groups of binary quadratic forms

- $\mathrm{Cl}(\Delta)$ with $\Delta < 0$ an integer
- Group elements : $f = \mathbf{a}x^2 + \mathbf{b}xy + \mathbf{c}y^2$ with discriminant $\Delta = b^2 - 4ac$

👍 Efficient algorithms for group operation: Shanks NATO NTA'89

👍 Public coin setup: $\Delta$ is known

👍 Hard problems are well studied, and harder than factoring

👎 Hard problems are not quantum-resistant

# Hard Problems in Class Goups

- Hard problems : DLog and computing the order of $\mathrm{Cl}(\Delta)$
- Sub exponential algorithm Hafner and McCurley, JoAMS'89
- Heuristic Complexity $L_{|\Delta|}[1/2, c + o(1)]$

  vs. factoring $L_{|N|}[1/3, c' + o(1)]$

$\rightsquigarrow$ Smaller parameters than for factoring

| Security | $\Delta$ **bits** | $N$ bits |
|---|---|---|
| 112 bits | **1348** | 2048 |
| 128 bits | **1827** | 3072 |
| 192 bits | **3598** | 7680 |
| 256 bits | **5971** | 15360 |

# CL Framework

Castagnos, Laguillaumie CT-RSA'15, Castagnos, Laguillaumie, Tucker ASIACRYPT'18

## Group with an easy DLog subgroup

- $q$ a prime
- $G = \langle g \rangle$ cyclic group of order $q \cdot s$ such that $\gcd(q, s) = 1$
- $F = \langle f \rangle$ subgroup of $G$ of order $q$
  - DLog is easy in $F$: given $u \in F$, find $m \in \mathbf{Z}/q\mathbf{Z}$ such that $u = f^m$
- $G^q = \langle h \rangle = \{x^q, x \in G\}$ subgroup of $G$ of order $s$,

$$G \simeq F \times G^q$$

- Hard to distinguish elements of $G^q$:

$$\{Z \hookleftarrow G\} \approx_c \{Z \hookleftarrow G^q\}$$

Hard Subgroup Membership Assumption (HSM)

$\rightarrow$ Encode m in $F$, hide it in $G$

# CL Framework

Castagnos, Laguillaumie CT-RSA'15, Castagnos, Laguillaumie, Tucker ASIACRYPT'18

## Group with an easy DLog subgroup

- $q$ a prime
- $G = \langle g \rangle$ cyclic group of order $q \cdot s$ such that $\gcd(q, s) = 1$
- $F = \langle f \rangle$ subgroup of $G$ of order $q$
  - DLog is easy in $F$: given $u \in F$, find $m \in \mathbf{Z}/q\mathbf{Z}$ such that $u = f^m$
- $G^q = \langle h \rangle = \{x^q, x \in G\}$ subgroup of $G$ of order $s$,

$$G \simeq F \times G^q$$

Sampling in $G^q$

- An upper bound for the order: $\tilde{s} > s$
- Distribution $\mathcal{D}_q$: uniform in $\{0, \ldots, 2^\sigma \tilde{s}\}$
- If $x \hookleftarrow \mathcal{D}_q$ then $h^x$ is close to uniform in $G^q$ (with distance $2^{-\sigma}$)

# CL Framework

Castagnos, Laguillaumie CT-RSA'15, Castagnos, Laguillaumie, Tucker ASIACRYPT'18

Group with an easy DLog subgroup

- $q$ a prime
- $G = \langle g \rangle$ cyclic group of order $q \cdot s$ such that $\gcd(q, s) = 1$
- $F = \langle f \rangle$ subgroup of $G$ of order $q$
  - DLog is easy in $F$: given $u \in F$, find $m \in \mathbf{Z}/q\mathbf{Z}$ such that $u = f^m$
- $G^q = \langle h \rangle = \{x^q, x \in G\}$ subgroup of $G$ of order $s$,

$$G \simeq F \times G^q$$

Instantiation of $G \subset \mathrm{Cl}(\Delta)$

- Choose $q$, sample $p$ prime, set $\Delta = -pq^3$
- Easy to sample $f$ and $h$ when we know $q$ and $p$
- Order of G is hard to compute !

# CL encryption scheme

Setup$(1^\lambda, q)$

- Run Gen$(1^\lambda, q)$
- Return the output $pp = (\tilde{s}, \mathcal{D}_q, f, h, G, G^q, F)$ //Order bound, distribution, group generators and repr.

KeyGen$(pp)$

- Pick $\mathsf{sk} \hookleftarrow \mathcal{D}_q$ and $\mathsf{pk} := h^{\mathsf{sk}}$
- Return $(\mathsf{pk}, \mathsf{sk})$

Enc$(\mathsf{pk}, m)$

- Pick $r \hookleftarrow \mathcal{D}_q$
- Return $c = (h^r, \mathsf{pk}^r f^m)$

Dec$(\mathsf{sk}, c = (c_1, c_2))$

- Compute $M = c_2 / c_1^{\mathsf{sk}}$
- Return DLog.Solve$(M)$

EvalScal$(\mathsf{pk}, c = (c_1, c_2), \alpha)$

- Compute $c_1' = c_1^\alpha$ and $c_2' = c_2^\alpha$
- Sample $r \hookleftarrow \mathcal{D}_q$
- Return $(c_1' \cdot h^r, c_2' \cdot \mathsf{pk}^r)$

EvalSum$(\mathsf{pk}, c = (c_1, c_2), c' = (c_1', c_2'))$

- Compute $c_1'' = c_1 \cdot c_1'$ and $c_2'' = c_2 \cdot c_2'$
- Sample $r \hookleftarrow \mathcal{D}_q$
- Return $(c_1'' \cdot h^r, c_2'' \cdot \mathsf{pk}^r)$

IND-CPA secure under the HSM assumption

# BICYCL 

- BICYCL Implements CryptographY in CLass groups
  Bouvier, Castagnos, Imbert, Laguillaumie, JoC'23
  https://ia.cr/2022/1466
- Efficient implementation of class group arithmetic
- CL and CL-based threshold protocols (Threshold-CL, Threshold-ECDSA)
- Appearences in the NIST Threshold Call (so far)
  - 2p ECDSA : TECLA
  - Th-ECDSA : THE CLASH, Trout++, Twig
- Open source, available at https://gite.lirmm.fr/crypto/bicycl

# CL vs Paillier

| | Encryption | | Decryption | | Ciphertext size | |
|---|---|---|---|---|---|---|
| Security level | Paillier | CL | Paillier | CL | Paillier | CL [1] |
| 112-bit sec. | 6 ms | 6 ms | **6** ms | 7 ms | 4.1 kbit | **2.8 kbit** |
| 128-bit sec. | 14 ms | **11 ms** | 14 ms | **13 ms** | 6.1 kbit | **3.5 kbit** |
| 192-bit sec. | 148 ms | **42 ms** | 148 ms | **57 ms** | 15.4 kbit | **6.5 kbit** |
| 256-bit sec. | 870 ms | **120 ms** | 876 ms | **149 ms** | 30.7 kbit | **10.5 kbit** |

- Paillier requires a trusted setup for MPC
- CL has a public coin setup and smaller ciphertexts

[1] compressed

# Outline

Threshold Linear Homomorphic Encryption

CL

Threshold CL

# Distributed Key Generation for Elgamal

Elgamal KeyGen: $\mathsf{sk} = x \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $\quad \mathsf{pk} \leftarrow g^{\mathsf{sk}}$

## DKG for Elgamal Pedersen EC'91

- Deal: party $i$
  - Samples $\alpha_i$, shares it: $P_i(X) = \alpha_i + r_{i,1}X + ... + r_{i,t}X^t$
  - Commits $g^{\alpha_i}, g^{r_{i,1}}, \ldots, g^{r_{i,t}} \rightarrow$ broadcast
  - Evaluates $\alpha_{i,j} := P_i(j) \rightarrow$ privately sends to party j
- Verify: party $i$
  - Checks received shares: $g^{\alpha_{i,j}} = \prod_{k=0}^{t} (g^{r_{j,k}})^{j^k}$ for each party $j$
  - Determines the set of qualified parties $\mathcal{Q}$
- Extract: party $i$
  - Computes $\sum_{j \in \mathcal{Q}} \alpha_{j,i} \rightsquigarrow$ secret sharing of implicit $\mathsf{sk} := \sum_{i \in \mathcal{Q}} \alpha_i$
  - Computes $\mathsf{pk} := \prod_{i \in \mathcal{Q}} g^{\alpha_i}$

$\rightarrow$ Adapt it to CL

# Distributed Key Generation for Elgamal

Elgamal KeyGen: $\mathsf{sk} = x \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $\quad \mathsf{pk} \leftarrow g^{\mathsf{sk}}$

## DKG for Elgamal Pedersen EC'91

- Deal: party $i$
  - Samples $\alpha_i$, shares it: $P_i(X) = \alpha_i + r_{i,1}X + ... + r_{i,t}X^t$
  - Commits $g^{\alpha_i}, g^{r_{i,1}}, \ldots, g^{r_{i,t}} \rightarrow$ broadcast
  - Evaluates $\alpha_{i,j} := P_i(j) \rightarrow$ privately sends to party j
- Verify: party $i$
  - Checks received shares: $g^{\alpha_{i,j}} = \prod_{k=0}^{t} (g^{r_{j,k}})^{j^k}$ for each party $j$
  - Determines the set of qualified parties $\mathcal{Q}$
- Extract: party $i$
  - Computes $\sum_{j \in \mathcal{Q}} \alpha_{j,i} \rightsquigarrow$ secret sharing of implicit $\mathsf{sk} := \sum_{i \in \mathcal{Q}} \alpha_i$
  - Computes $\mathsf{pk} := \prod_{i \in \mathcal{Q}} g^{\alpha_i}$

$\rightarrow$ Adapt it to CL

# Distributed Key Generation for Elgamal

Elgamal KeyGen: $\mathsf{sk} = x \hookleftarrow \mathbf{Z}/q\mathbf{Z}$ $\quad \mathsf{pk} \leftarrow g^{\mathsf{sk}}$

## DKG for Elgamal Pedersen EC'91

- Deal: party $i$
  - Samples $\alpha_i$, shares it: $P_i(X) = \alpha_i + r_{i,1}X + ... + r_{i,t}X^t$
  - Commits $g^{\alpha_i}, g^{r_{i,1}}, \ldots, g^{r_{i,t}} \rightarrow$ broadcast
  - Evaluates $\alpha_{i,j} := P_i(j) \rightarrow$ privately sends to party j
- Verify: party $i$
  - Checks received shares: $g^{\alpha_{i,j}} = \prod_{k=0}^{t} (g^{r_{j,k}})^{j^k}$ for each party $j$
  - Determines the set of qualified parties $\mathcal{Q}$
- Extract: party $i$
  - Computes $\sum_{j \in \mathcal{Q}} \alpha_{j,i} \rightsquigarrow$ secret sharing of implicit $\mathsf{sk} := \sum_{i \in \mathcal{Q}} \alpha_i$
  - Computes $\mathsf{pk} := \prod_{i \in \mathcal{Q}} g^{\alpha_i}$

$\rightarrow$ Adapt it to CL

# Adapting DKG to CL

CL.KeyGen($pp$): $\mathsf{sk} \hookleftarrow \mathcal{D}_q$; $\mathsf{pk} \leftarrow h^{\mathsf{sk}}$

## DKG for CL

Braun,Damgård, Orlandi, C'23; Braun, Castagnos, Damgård, Laguillaumie, Melissaris, Orlandi, Tucker, SCN'24 `https://ia.cr/2024/717`

- Elgamal: computations modulo the known order of $g$
- CL: Unknown order
  - $\rightsquigarrow$ Use Shamir's secret sharing over $\mathbf{Z}$: $P_i(X) = n! \cdot \alpha_i + r_{i,1}X + ... + r_{i,t}X^t$
  - $\rightsquigarrow$ Secret sharing reconstruction scaled by an additional factor $n!$
  - $\rightsquigarrow$ implicit $sk = (n!)^2 \cdot \sum_{i \in \mathcal{Q}} \alpha_i$
- Commitments in $G \subset \mathsf{CL}(\Delta)$, not publicly checkable
  - $\rightsquigarrow$ Additional ZK-proof for CL-DLog during the verification phase

# Distributed Decryption for CL

CL.Dec($\mathsf{sk}, c = (c_1, c_2)$): $M \leftarrow c_2 / c_1^{\mathsf{sk}}$ $m \leftarrow \text{DLog.Solve}(M)$

## Distributed computation of $c_1^{sk}$

$\gamma_i$: secret share of $sk$

- Partial dec.: qualified party $i$ computes and publishes $\omega_i = c_1^{(n!)^2 \cdot \gamma_i}$ (+ZK-proof)
- Verify ZK-proofs, determine the set S of parties for this decryption
- Combine the results using Lagrange coefficients in the exponent
  - Reconstruct $W = \prod_{P_j \in S} {\omega_i}^{L_j(0)} = c_1^{(n!)^2 \cdot sk}$
  - Compute DLog.Solve$(c_2^{(n!)^2} / W) \rightsquigarrow (n!)^2 \cdot m$
  - Defined modulo the known prime $q \rightsquigarrow$ We can recover $m$

# Threshold CL performance

| $n$ | 3 | 5 | 7 | 10 | 20 | 50 | 100 | 200 |
|---|---|---|---|---|---|---|---|---|
| Execution (ms) | | | | | | | | |
| Keygen | 77 | 129 | 189 | 280 | 675 | 2 776 | 9 500 | 37 000 |
| Decrypt | 100 | 121 | 122 | 127 | 147 | 221 | 373 | 862 |
| Communication per party (kB) | | | | | | | | |
| Keygen | 1.1 | 1.6 | 2.1 | 2.8 | 5.5 | 14.7 | 35 | 96 |
| Decrypt | 0.41 | 0.55 | 0.61 | 0.66 | 0.74 | 0.82 | 0.89 | 1.02 |

Threshold-CL (static security) benchmarks from BICYCL. $t = n/2 - 1$
Computational sec. $\kappa = 128$ bits, statistical sec. $\sigma = 40$ bits.