Multilinear Maps and Their Applications

Mark Zhandry – Stanford University

Diffie-Hellman Key Exchange

Exchange keys over a public channel:

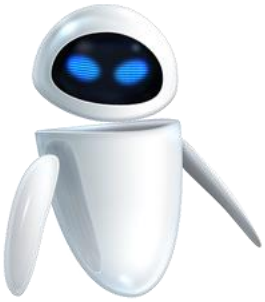
- Public group $\mathbb{G}$, generator g, order p

g^a

g^b

$a \xleftarrow{\$} \mathbb{Z}_p$

$K = (g^b)^a = g^{ab}$



$b \xleftarrow{\$} \mathbb{Z}_p$

$K = (g^a)^b = g^{ab}$

$g, g^a, g^b \xrightarrow{?} g^{ab}$

(Potential) Hard Problems in Groups

- Discrete Log (DL):

$$g, g^a \longrightarrow a$$

- Computational Diffie-Hellman (CDH):

$$g, g^a, g^b \longrightarrow g^{ab}$$

- Decisional Diffie-Hellman (DDH):

$$g, g^a, g^b, g^c \longrightarrow c = ab?$$

- Many Others:
  - Decision Linear (DLIN):

$$g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \longrightarrow c = a + b?$$

Uses of Diffie-Hellman

- Two party key exchange
- Encryption
- Signatures
- …

3-Way Diffie-Hellman?

g^a

g^b

$a \xleftarrow{\$} \mathbb{Z}_p$

$b \xleftarrow{\$} \mathbb{Z}_p$

$K = g^{abc} = ???$

g^c

$c \xleftarrow{\$} \mathbb{Z}_p$

3-Way Diffie-Hellman

Problem: Need way to multiply g^b and g^c

Solution [Joux'00]: Use bilinear maps

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_2$$

$$e(g^a, g^b) = g_2^{ab}$$

- Bilinear group: group with bilinear map

3-Way Diffie-Hellman?

g^a

g^b

$a \xleftarrow{\$} \mathbb{Z}_p$

$b \xleftarrow{\$} \mathbb{Z}_p$

$K = g_2^{abc} = e(g^b, g^c)^a$

g^c

$c \xleftarrow{\$} \mathbb{Z}_p$

Potential Hard Problems in Bilinear Groups

- DL, CDH, DLIN
- DDH? $e(g^a, g^b) \stackrel{?}{=} e(g, g^c)$
- Bilinear DDH:

$$g, g^a, g^b, g^c, g_2^d \longrightarrow d = abc?$$

- Many Others
  - Bilinear Diffie-Hellman Exponent
  - Subgroup Decision
  - …

Uses of Bilinear Maps

- Identity-Based Encryption
- Broadcast Encryption w/ short ciphertexts
- Traitor Tracing w/ short ciphertexts
- Short Signatures
- Threshold Signatures
- Somewhat Homomorphic Encryption
- …

4-Way Diffie Hellman?

g^a

$a \xleftarrow{\$} \mathbb{Z}_p$

g^b

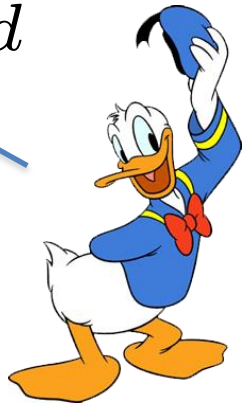
$b \xleftarrow{\$} \mathbb{Z}_p$

$K = ?$

g^c

$c \xleftarrow{\$} \mathbb{Z}_p$

g^d



$d \xleftarrow{\$} \mathbb{Z}_p$

Multilinear Maps

Many groups: $\mathbb{G}_1, \mathbb{G}_2, \ldots \mathbb{G}_\kappa$

- Generators g_i

Source group: $\mathbb{G} = \mathbb{G}_1,\ g = g_1$

Pairing: $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \longrightarrow \mathbb{G}_{i+j}$
$e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$ $(i + j \leq \kappa)$

- Often write $e = e_{i,j}$

Gives multilinear map: $e : \mathbb{G}^\kappa \longrightarrow \mathbb{G}_\kappa$
$e(g^{a_1}, \cdots, g^{a_\kappa}) = g_\kappa^{a_1 \cdots a_\kappa}$

Potential Hard Problems in Multilinear Groups

- DL, CDH, generalization of DLIN
- Multilinear DDH:

$$g, g^{a_1}, \ldots, g^{a_{\kappa+1}}, g_\kappa^c \longrightarrow c \stackrel{?}{=} \prod_{i=1}^{\kappa+1} a_i$$

- ML-CDH for all $\kappa' \leq \kappa$
  - ML-DDH easy for all $\kappa' < \kappa$

- Many others:
  - Subgroup Decision
  - Multilinear DH Exponent

Potential Applications

Or: Imagine what we could do…

N-Way Key Exchange

$a \xleftarrow{\$} \mathbb{Z}_p$

g^a

g^b

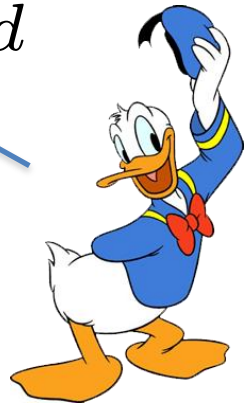
$b \xleftarrow{\$} \mathbb{Z}_p$

$K = g_3^{abcd}$

g^c

$c \xleftarrow{\$} \mathbb{Z}_p$

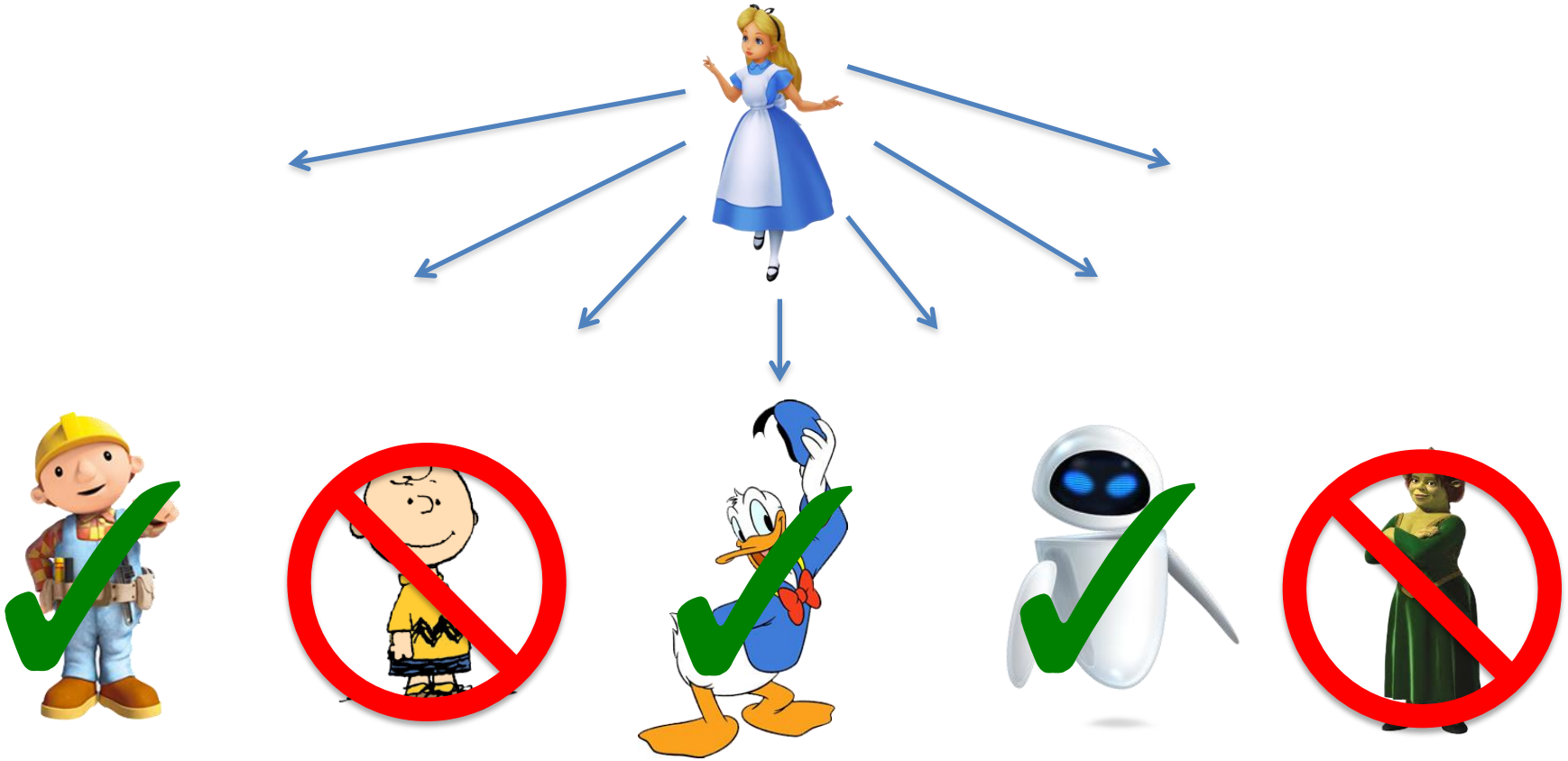
g^d



$d \xleftarrow{\$} \mathbb{Z}_p$

Broadcast Encryption

- Alice wants to broadcast a message
- Only a subset of players should decrypt



- Will build via constrained PRFs

PRFs

Keyed functions that look like random functions

$$f : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$$

All or Nothing:

- Given k, can eval $f_k(x)$ at all x
- Without k, $f_k(\cdot)$ indistinguishable from random

Constrained PRFs [BW'13]

Given set S of inputs, give "constrained key":

$$k_S \leftarrow \texttt{constrain}(k, S)$$

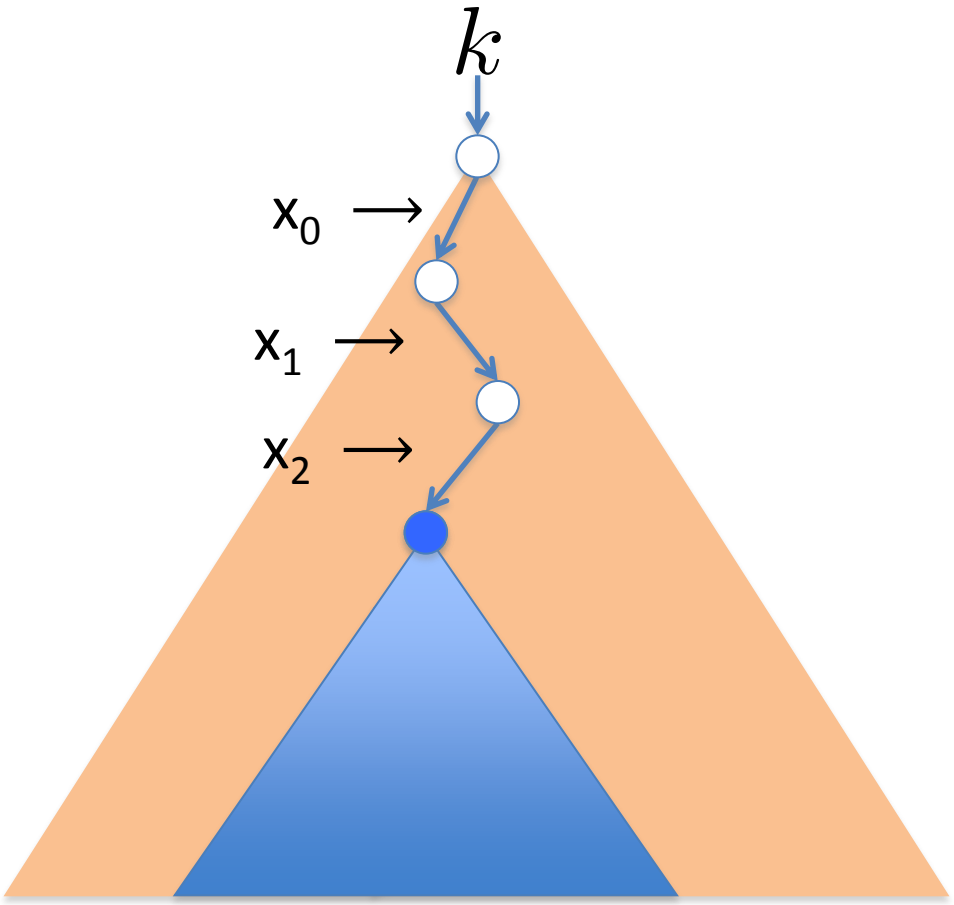
k_S can compute $f_k(x)$ on all points $x \in S$:

$$\texttt{eval}(k_S, x) = \begin{cases} f_k(x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

Goal: allow interesting sets S

Example: GGM

Constrained keys = values of nodes



Constrained sets = sets with same prefix

Other Possible Set Systems

Left/Right: $\mathcal{X} = \mathcal{W}^2$

- Left sets: (x_0, y) for fixed x_0
- Right sets: (x, y_0) for fixed y_0

Bit-fixing: $\mathcal{X} = \{0,1\}^n$

- Sets correspond to $\mathbf{v} \in \{0,1,?\}^n$
- Can eval at all x that agree with $\mathbf{v}$ ($?$ wildcard)

  Example: $\mathbf{v} = (0,?,?,1) \rightarrow S = \{0001, 0011, 0101, 0111\}$

Circuit Predicates

Bit-Fixing PRF Construction

Use $\kappa = n + 1$ multilinear map

Setup:

- Choose random $\{d_{i,\beta}\}_{i \in [n], \beta \in \{0,1\}}$
- Choose random α
- Secret key: $(\alpha, \{d_{i,\beta}\}_{i \in [n], \beta \in \{0,1\}})$

Function:

$$f_k(x) = g_\kappa^{\alpha \prod_i d_{i,x_i}}$$

Bit-Fixing PRF Construction

$$f_k(x) = g_\kappa^{\alpha \prod_i d_{i,x_i}}$$

Constrain:

- Input $\mathbf{v} \in \{0,1,?\}^n$
- Let $V = \{i \in [n] : \mathbf{v}_i \neq ?\}$
- $k'_{\mathbf{v}} = (g_{1+|V|})^{\alpha \prod_{i \in V} d_{i,\mathbf{v}_i}}$
- $k_{\mathbf{v}} = (k'_{\mathbf{v}}, \{g^{d_{i,\beta}}\}_{i \notin V, \beta \in \{0,1\}})$

Bit-Fixing PRF Construction

$$f_k(x) = g_\kappa^{\alpha \prod_i d_{i,x_i}}$$

$$k'_{\mathbf{v}} = (g_{1+|V|})^{\alpha \prod_{i \in V} d_{i,\mathbf{v}_i}}$$

Eval:

- $T = (g_{n-|V|})^{\prod_{i \notin V}(d_{i,x_i})}$
- Pair with $k'_{\mathbf{v}}$ to get output

Broadcast Encryption from Bit-Fixing PRFs

Setup:

- Generate a Bit-Fixing PRF f with key k
- For each player $i \in [n]$, compute:

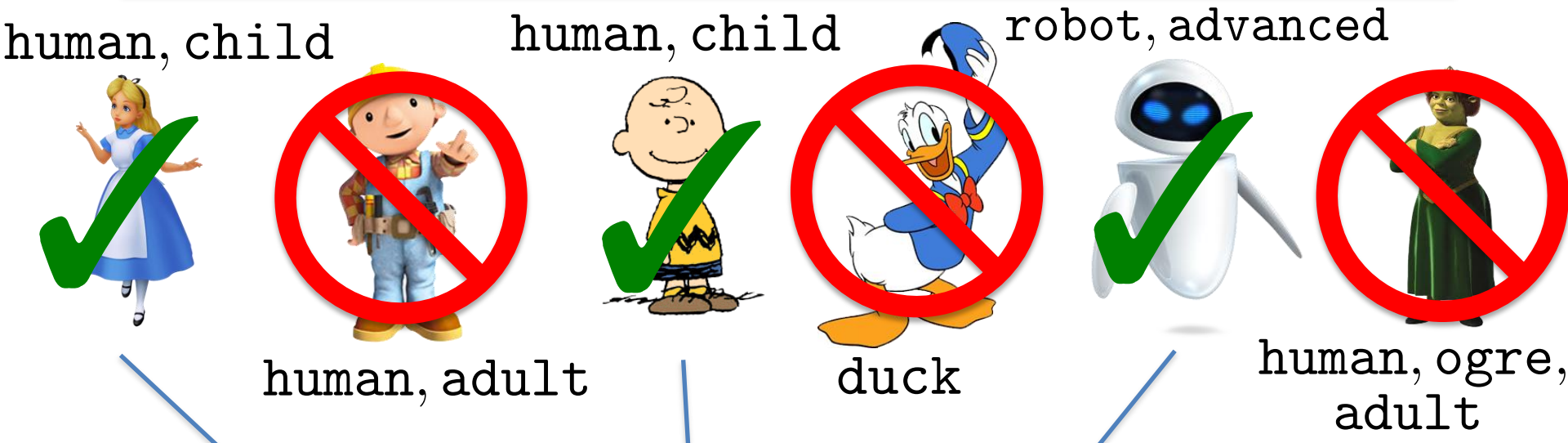
$$k_i \leftarrow \texttt{constrain}(k, \mathbf{v})$$

where $\mathbf{v}_i = 1$, $\mathbf{v}_j = ?$ for $j \neq i$

Encrypt to a subset $S \subseteq [n]$ of players:

- Let $k_S = f(S)$
- Use symmetric cipher with key k_S

Policy-Based Key Agreement



$$\texttt{robot} \vee (\texttt{human} \wedge \texttt{child})$$

Shared secret key

Build from constrained PRFs for circuit predicates

Other Applications of Multilinear Maps

- Attribute-Based Encryption
- Witness Encryption
- Obfuscation
- Functional Encryption
- …

Rest of Talk

Two recent candidates for multilinear maps

- From ideal lattices
- Over the integers

Not true multilinear maps

- Randomized
- Noisy

May still be used in many applications

Relaxation: Graded Encodings

Scalar a → Level 0 encoding of α

g^a → Level 1 encoding of α

g_2^a → Level 2 encoding of α

…

Graded encoding schemes: encoding not unique

- Ring $R, \alpha \in R$

$S_i^{(\alpha)}$: set of level i encodings of α

Relaxation: Graded Encodings

Requirements:

- Add same level encodings

$$a \in S_i^{(\alpha)} \qquad b \in S_i^{(\beta)}$$

$$\mathrm{add}(a, b) \in S_i^{(\alpha+\beta)}$$

- Multiply encodings

$$a \in S_i^{(\alpha)} \qquad b \in S_j^{(\beta)}$$

$$\mathrm{mult}(a, b) \in S_{i+j}^{(\alpha\beta)}$$

(as long as $i + j \leq \kappa$)

Pairing Equivalent:

$$a, b \to a + b$$

$$g^a, g^b \to g^a g^b = g^{a+b}$$

$$g_2^a, g_2^b \to g_2^a g_2^b = g_2^{a+b}$$

$$a, b \to ab$$

$$a, g^b \to (g^b)^a = g^{ab}$$

$$g^a, g^b \to e(g^a, g^b) = g_2^{ab}$$

The GGH Construction

Notation

$R = \mathbb{Z}[X]/(X^n + 1)$

$\mathbb{K} = \mathbb{Q}[X]/(X^n + 1)$

$R' = R/qR$

$[\mathbf{u}]_q$: reduce mod q

$\langle \mathbf{g} \rangle$: principle ideal generated by $\mathbf{g} \in R$

Properties:

- $R \longleftrightarrow \mathbb{Z}^n, \quad R' \longleftrightarrow \mathbb{Z}_q^n$
- $\mathbf{u}, \mathbf{v}$ "short" → $\mathbf{u} + \mathbf{y}$ $\mathbf{uv}$ "short"

The GGH Construction

- $\mathbf{g} \in R$ "short", secret, $\mathbf{g}^{-1} \in \mathbb{K}$ "short"
- $\mathcal{I} = \langle \mathbf{g} \rangle$
- $T = R/\mathcal{I} \longleftrightarrow \{\mathbf{e} + \mathcal{I} : \mathbf{e} \in R\}$
- $\mathbf{z} \in R'$ secret, not short
- Level i encoding of $\mathbf{e} + \mathcal{I} \in T$:

$$\frac{\mathbf{c}}{\mathbf{z}^i} \in R_q \qquad \mathbf{c} \in \mathbf{e} + \mathcal{I}, \text{ "short"}$$

$$S_i^{\mathbf{e}+\mathcal{I}} = \left\{ \frac{\mathbf{c}}{\mathbf{z}^i} : \mathbf{c} \in \mathbf{e} + \mathcal{I}, \|\mathbf{c}\| \text{ small} \right\}$$

Encoding Operations

- Addition:

$$\mathbf{u}_1 = \frac{\mathbf{c}_1}{\mathbf{z}^i} \in S_i^{(\mathbf{e}_1+\mathcal{I})} \quad \mathbf{u}_2 = \frac{\mathbf{c}_2}{\mathbf{z}^i} \in S_i^{(\mathbf{e}_2+\mathcal{I})}$$

$$\mathbf{u}_1 + \mathbf{u}_2 = \frac{\mathbf{c}_1 + \mathbf{c}_2}{\mathbf{z}^i} \in S_i^{(\mathbf{e}_1+\mathbf{e}_2+\mathcal{I})}$$

Proof:

$$\mathbf{c}_1 + \mathbf{c}_2 \in (\mathbf{e}_1 + \mathbf{e}_2) + \mathcal{I}$$

$\mathbf{c}_1 + \mathbf{c}_2$ “short”

Encoding Operations

- Multiplication:

$$\mathbf{u}_1 = \frac{\mathbf{c}_1}{\mathbf{z}^i} \in S_i^{(\mathbf{e}_1+\mathcal{I})} \quad \mathbf{u}_2 = \frac{\mathbf{c}_2}{\mathbf{z}^j} \in S_j^{(\mathbf{e}_2+\mathcal{I})}$$

$$\mathbf{u}_1\mathbf{u}_2 = \frac{\mathbf{c}_1\mathbf{c}_2}{\mathbf{z}^{i+j}} \in S_{i+j}^{(\mathbf{e}_1\mathbf{e}_2+\mathcal{I})}$$

Proof:

$\mathbf{c}_1\mathbf{c}_2 \in (\mathbf{e}_1\mathbf{e}_2) + \mathcal{I}$

$\mathbf{c}_1\mathbf{c}_2$ “short”

Generating Level 0 Encodings

Level 0 encoding of $\mathbf{e} + \mathcal{I}$: short $\mathbf{c} \in \mathbf{e} + \mathcal{I}$

Problem: can't encode coset w/o knowing $\mathbf{g}$

Resolution: sample coset by sampling short rep

Fact: Sample "short" $\mathbf{c}$ from appropriate distribution → coset $\mathbf{c} + \mathcal{I}$ close to uniform

Moving to Higher Levels

Need operation $\mathbf{c} \rightarrow \frac{\mathbf{c}'}{\mathbf{z}}$ where $\mathbf{c}, \mathbf{c}' \in \mathbf{e} + \mathcal{I}$

Problem: $\mathbf{z}$ is secret

Solution: publish level 1 encoding of $1 + \mathcal{I}$

$$\mathbf{y} = \left[\frac{\mathbf{a}}{\mathbf{z}}\right]_q \quad \mathbf{a} \in 1 + \mathcal{I}, \text{“short”}$$

To move to level 1:

$$\mathbf{c} \rightarrow [\mathbf{yc}]_q = \left[\frac{\mathbf{ac}}{\mathbf{z}}\right]_q$$

Moving to Higher Levels

Insecure: $[\mathbf{yc}]_q \to \mathbf{c}$ by dividing by $\mathbf{y}$

Solution: rerandomize

- Publish many level 1 encodings of 0:

$$\mathbf{x}_i = \left[\frac{\mathbf{b}_i}{\mathbf{z}}\right]_q \qquad \mathbf{b}_i \in \mathcal{I}, \text{“short”}$$

To move to level 1:

$$\mathbf{c} \to [\mathbf{yc} + \sum_i r_i \mathbf{x}_i]_q = \left[\frac{\mathbf{ac} + \sum_i r_i \mathbf{b}_i}{\mathbf{z}}\right]_q$$

$$r_i \in \mathbb{Z}, \text{“small”}$$

Testing for Equality

Need to be able to test equality

- Suffices to test if level κ encoding encodes 0

Solution: publish "zero test" parameter

$$\mathbf{p}_{zt} = \left[\frac{\mathbf{h}\mathbf{z}^{\kappa}}{\mathbf{g}} \right]_q$$

$\mathbf{h}$ "somewhat small"

Test if $[\mathbf{p}_{zt}\mathbf{u}]_q$ is "small"

Testing for Equality

$$[\mathbf{p}_{zt}\mathbf{u}]_q = \left[\frac{\mathbf{h}\mathbf{z}^{\kappa}}{\mathbf{g}}\frac{\mathbf{c}}{\mathbf{z}^{\kappa}}\right]_q = \left[\mathbf{h}\frac{\mathbf{c}}{\mathbf{g}}\right]_q$$

If $\mathbf{u}$ encodes 0: $\mathbf{c} \in \mathcal{I}$

$$\frac{\mathbf{c}}{\mathbf{g}} = \mathbf{c}\mathbf{g}^{-1}$$ (Multiplication over $\mathbb{K}$)

$$[\mathbf{p}_{zt}\mathbf{u}]_q = [\mathbf{h}\mathbf{c}\mathbf{g}^{-1}]_q = \mathbf{h}\mathbf{c}\mathbf{g}^{-1}$$ "short"

Testing for Equality

$$[\mathbf{p}_{zt}\mathbf{u}]_q = \left[\frac{\mathbf{h}\mathbf{z}^{\kappa}}{\mathbf{g}}\frac{\mathbf{c}}{\mathbf{z}^{\kappa}}\right]_q = \left[\mathbf{h}\frac{\mathbf{c}}{\mathbf{g}}\right]_q$$

If $\mathbf{u}$ encodes non-zero: $\mathbf{c} \notin \mathcal{I}$

Thm [GGH]: If $\mathbf{c} \notin \mathcal{I}$, then $\left[\mathbf{h}\frac{\mathbf{c}}{\mathbf{g}}\right]_q$ is large w.h.p.

Extraction

Each party needs to agree on same value

- But have different encoding of same element

Solution: Use zero-test parameter

- If $\mathbf{u}, \mathbf{u}'$ encode same value,
  $[\mathbf{p}_{zt}\mathbf{u}]_q - [\mathbf{p}_{zt}\mathbf{u}']_q = [\mathbf{p}_{zt}(\mathbf{u} - \mathbf{u}')]_q$ is "short"

- $[\mathbf{p}_{zt}\mathbf{u}]_q, [\mathbf{p}_{zt}\mathbf{u}']_q$ agree on most-significant bits

Extraction

To extract at level κ:

- Collect most-significant bits of $[\mathbf{p}_{zt}\mathbf{u}]_q$
- Apply strong randomness extractor to get uniform bit string

What needs to be a secret?

- $\mathbf{z}$: otherwise DL is easy
- $\mathbf{g}$: compute $\mathbf{p}'_{zt} = [\mathbf{g}\mathbf{p}_{zt}]_q = [\mathbf{h}\mathbf{z}^\kappa]_q$

  Given level 1 encoding $\mathbf{u} = \dfrac{\mathbf{c}}{\mathbf{z}}$

  Compute $\mathbf{v}_1 = [\mathbf{p}'_{zt}\mathbf{u}\mathbf{y}^{\kappa-1}]_q = \mathbf{h}\mathbf{a}^{\kappa-1}\mathbf{c}$

  $$\mathbf{v}_2 = [\mathbf{p}'_{zt}\mathbf{y}^\kappa]_q = \mathbf{h}\mathbf{a}^\kappa$$

  No $\mathbf{g}$, so can "divide mod $\mathcal{I}$"

  – Obtain $\mathbf{c}' \in \mathbf{c} + \mathcal{I}$, "short"

What needs to be a secret?

- $\mathbf{h}$: compute $\mathbf{p}'_{zt} = [\mathbf{p}_{zt}/\mathbf{h}]_q = [\mathbf{z}^\kappa/\mathbf{g}]_q$

  Pick randomizer

$$\mathbf{x}_i = \left[\frac{\mathbf{b}_i}{\mathbf{z}}\right]_q \qquad \mathbf{b}_i = \mathbf{g}\mathbf{b}'_i$$

$\mathbf{b}'_i$ "short"

  Compute

$$\mathbf{p}''_{zt} = [(\mathbf{p}'_{zt})^2\mathbf{x}_i]_q = [\mathbf{b}'_i\mathbf{z}^{2\kappa}/\mathbf{g}]_q$$

  Now we have level 2k zero tester!

  → Can solve MLDDH

Security of GGH

- No security proof from standard assumptions
  - Instead: extensive cryptanalysis
- Supposed hard problems:
  - Discrete Log
  - Multilinear DDH
- Easy problems:
  - Decision Linear
  - Subgroup Decision

Efficiency of GGH

- Parameterized by security λ, level κ
- All encodings represented as elements in $\mathbb{Z}_q^n$

- For functionality, need (at minimum)

$$\log q \geq O(\kappa\lambda + \kappa \log n)$$

- For security, need $\log q < n/\lambda$
  - Implies $n \geq \tilde{O}(\kappa\lambda^2)$

- Size of encodings: $\tilde{O}(\kappa^2\lambda^3)$

Efficiency of GGH

- Size of encodings: $\tilde{O}(\kappa^2 \lambda^3)$

- Size of public parameters:
  - Level 1 encoding of 1
  - m level 1 encodings of 0

    ($m > O(n \log q)$ for rerandomization)
  - Zero tester

  Total public parameter size: $\tilde{O}(k^4 \lambda^6)$
    - Even larger for some applications

The CLT Construction

The CLT Construction

$R = \mathbb{Z}^n$, component-wise add/mult

Let $\mathbf{p} = (p_1, ..., p_n) \in R$ vector of primes

$R' = R/\mathbf{p}R$

$\mathbf{g} \in R$ "short", secret vector of primes

$\mathcal{I} = \langle \mathbf{g} \rangle$

$T = R/\mathcal{I} \longleftrightarrow \{\mathbf{e} : 0 \leq e_i \leq g_i\}$

$\mathbf{z} \in R'$ secret, not short

Over the Integers

Let $q = \prod p_i$

CRT isomorphism:

$$R' = \mathbb{Z}^n / \mathbf{p}\mathbb{Z}^n \longleftrightarrow \mathbb{Z}_q$$

Apply to scheme:

$z \in \mathbb{Z}_q$ random

Level i encoding of $\mathbf{e} = (e_1, ..., e_n)$:

$$\frac{c}{z^i} \in \mathbb{Z}_q \text{ s.t. } \quad c = r_j g_j + e_j \pmod{p_i}$$

$r_j g_j + e_j$ small

Secrets?

Need same secrets as GGH: g, z, h

What about the primes?

- Factorization of q known → 1D problem
- Look at what happens mod p
- GGH zero tester, encodings of 0, 1:

$$p_{zt} = \frac{hz^{\kappa}}{g} \pmod{p} \qquad x_{\ell} = \frac{r_{\ell} g}{z} \pmod{p}$$

$$y = \frac{1 + sg}{z} \pmod{p}$$

Secrets?

Combine:

$$[x_\ell^i y^{\kappa-i} p_{zt}]_p = [h r_\ell^i (1+sg)^{\kappa-i} g^{i-1}]_p$$
$$= h r_\ell^i (1+sg)^{\kappa-i} g^{i-1}$$

Compute for many i, ℓ, GCD → h

Compute for many $i \geq 2, \ell$, GCD → hg → g

From h, g easy to compute z

For security, must keep primes secret!

Other Changes

Keeping primes secret introduces several issues:

- Generating level 0 encodings

  Must generate integer c such that

  $$c \pmod{p_j}$$ is short

  Cannot sample without knowing p_j!

  Solution: publish many level 0 encodings w_ℓ of random values

  – To sample, take random subset sums

Other Changes

Keeping primes secret introduces several issues:

- Zero testing:

  GGH zero tester:

$$p_{zt} = \frac{h_j z^\kappa}{g_j} \pmod{p_j}$$

  Level κ encoding of 0:

$$u = \frac{r_j g_j}{z^k} \pmod{p_j}$$

Zero Testing

Multiply GGH zero tester with encoding of 0:

$$p_{zt} u = h_j r_j \pmod{p_j}$$

Product is "short" mod p_j, but we can't test!

Instead, want product to be "short" mod q

CRT Isomorphism

$$c \longrightarrow c_j = c \bmod p_j$$

$$c_j \longrightarrow c = \left[\sum_j c_j \left(\prod_{j' \neq j} p_{j'} (p_{j'}^{-1} \bmod p_j) \right) \right] \bmod q$$

Coefficient of c_j >> q

- Small c_j do not give small c

Need to cancel out some the coefficient

Zero Testing

Solution: new zero tester

$$p_{zt} = \frac{h_j z^k}{g_j} \left(\prod_{j' \neq j} p_{j'} \right) \pmod{p_j}$$

Multiply with encoding of zero:

$$p_{zt} u = h_j r_j \prod_{j' \neq j} p_{j'} \pmod{p_j}$$

CRT: $p_{zt} u = \sum_j h_j r_j \prod_{j' \neq j} p_{j'} \pmod{q}$

Zero Testing

$$p_{zt}u = \sum_j h_j r_j \prod_{j' \neq j} p_{j'} \pmod q$$

$$|p_{zt}u| \approx n\,|h|\,|r|\,p^{n-1} << p^n \approx q$$

Thm [CLT]: If u does not encode 0, then $|p_{zt}u| \approx q$ whp

Security of CLT

- Just like GGH, no security proof from standard assumptions

- Supposed hard problems:
  - Discrete Log
  - Multilinear DH
  - Decision Linear?
  - Subgroup Decision?

Efficiency of CLT

All encodings elements of $\mathbb{Z}_q$

Size of encodings same as GGH: $\tilde{O}(\kappa^2 \lambda^3)$

Public params:

- Asymptotically same: $\tilde{O}(k^4 \lambda^6)$
- CLT offer some heuristics to reduce size

Open Problems

- From standard assumptions
- Remove secrets
  - Necessary to remove trusted setup from key exchange
  - How to remove zero tester?
- More Efficient?