Introduction to Cryptographic Multilinear Maps

Sanjam Garg, Craig Gentry, Shai Halevi
IBM T.J. Watson Research Center

Multilinear Maps (MMAPs)

- A technical tool
  - Think "trapdoor-permutations" or "smooth-projective-hashing", or "randomized-encoding"
  - More a technique than a single primitive
    - Several different variants, all share the same core properties but differ in details
- Extension of bilinear maps [J00,SOK00,BF01]
  - Bilinear maps are extensions of DL-based crypto
  - Took the crypto world by storm in 2000, used in dozens of applications, hundreds of papers
  - Applications from IBE to NIZK and more

DL/DDH and Bilinear Maps

- Why is DDH such a "gold mine"?
  - You can take values a_i and "hide them" in g^{a_i}
  - Some tasks are still easy in this representation
    - Can compute any linear/affine function of the a_i's, and check if $a_i = 0$
  - Other tasks are seemingly hard
    - E.g., computing/checking quadratic functions
- Bilinear maps are similar: we can compute quadratics, while cubics seem hard
    - Turns out to be even more useful

Why Stop at Two?

- Can we find groups that would let us compute cubics but not 4^{th} powers?
  - Or in general, upto degree k but no more?

➔ Cryptographic multilinear maps (MMAPs)
  - Even more useful than bilinear

- [Boneh-Silverberg'03] explored some applications of MMAPs
  - Also argued that they are unlikely to be constructed similarly to bilinear maps

The [GGH'13] Approach

- An "approximate" cryptographic MMAPs
  - Degree-k functions, zero-test are easy
  - Some degree-$(k+1)$ functions seem hard
  - Enabling many new applications
- Built using "FHE techniques"
  - From a variant of the NTRU HE scheme
- Another construction in [CLT'13]
  - Using a variant of "HE over integers" instead
  - All degree-$(k+1)$ functions seem hard

This Talk

- An overview of [GGH'13]
  - Some details
  - Which degree-$(k+1)$ functions are easy/hard
    - Source- vs. target-group assumptions
- Examples of using it:
  - $(k+1)$-partite key exchange [J00,BS03]
  - Witness encryption [GGSW'13]
  - Full domain hash [FHPS'13, HSW'13]
  - Obfuscation (just a hint)

THE [GGH'13] CONSTRUCTION

"Somewhat Homomorphic" Encryption (SWHE) vs. MMAPs

SWHE

- Encrypting $\mathrm{c}_a = \mathrm{E}(a)$
- ✓ Computing low-deg polynomials of the c_a's is easy
- ? Fuzzy threshold for easy vs. hard?
- ✘ Cannot test anything
  - But if you have skey you can recover a itself

MMAPs

- "Encoding" $e_a = g^a$
- ✓ Computing low-deg polynomials of the e_a's is easy
- ✓ Sharp threshold for easy vs. hard
- ✓ Can test for zero

Main Ingredient: Testing for Zero

- To be useful, we must be able to test if two degree-k expressions are equal
  - Using homomorphism, that's the same as testing if a degree-k expression equals zero
- Our approach: augment a SWHE scheme with a "handicapped" secret key
  - Can test if a ciphertext decrypts to zero, but cannot decrypt arbitrary cipehrtexts
    - Assuming that the plaintext-space is large
  - Called a "*zero-test parameter*"

A Few Words About Levels

- A "level-k" ciphertext encrypts a degree-k expression
  - Fresh cipehrtexts, c_a=Enc(a), are at level 1
  - $Level(c \boxtimes c') = Level(c) + Level(c')$
  - $Level(c \boxplus c') = \max\{Level(c), Level(c')\}$
- Contemporary SWHE schemes are "naturally leveled"
  - Often a ciphertext in these schemes would be tagged with its level

A Few Words About Levels (2)

- A zero-test parameter that works for all levels, would give a "black-box field"
  - Could be useful, but it's not MMAPs
  - Also we don't know how to get one
- Our zero-test parameter only works for ciphertexts at one particular level k
  - The zero-test level is a parameter, equal to the multi-linearity degree that we want to implement

Our Goal ("approximate MMAPs")

a_i's from some large finite field/ring

k-Graded Encoding Scheme

- KeyGen(k): Generating public parameters
- Encode: level-1 encoding of plaintext a_i's
  - Plaintext a_i's themselves are considered "level-0"
  - Encoding can be randomized
- Arithmetic: addition & multiplication
  - $Level(c \boxtimes c') = Level(c) + Level(c')$
  - $Level(c \boxplus c') = \max\{Level(c), Level(c')\}$
- Zero-test: does c encode 0?
  - Only works for level-k encoding

Some Variations

- Can extract "random canonical representation" of a from any level-k encoding of a
- Can only encode *random* a_i's, not specific ones
- KeyGen outputs a matching secret key
  - Secret key may be needed for encoding
- Encoding can be re-randomizable
  - Given any level-i encoding of a, output a random level-i encoding of the same a
- More complicated level structure than just 0,1,2, …
  - E.g., levels are vectors, with partial ordering
  - Yields an extension of "asymmetric maps"

Overview of [GGH'13]

- Start from an NTRU-like SWHE scheme
  - Semantic-security under some "reasonable assumptions"
- Add zero-test parameter
  - Some things that were hard now become easy
  - Other things are still seemingly hard
    - But hardness assumptions are stronger, uglier
  - Separating hard from easy is challenging

Starting From NTRU-like SWHE

- All ops are in some polynomial rings
  - $R = Z[X]/F(X),\ R_q = R/qR$
- Secret key is $g, z \in R_q$
  - g is short $(|g| \ll q),\ z$ is random in R_q
  - Plaintext elements are from $R_g = R/gR$
- An encryption of a is $c_a = [e_a/z]_q$
  - $|e_a| \ll q$ and $e_a = a\ (mod\ g)$
- To decrypt set $a \leftarrow [c_a \cdot z]_q\ mod\ g$

In NTRU $g = 3$

Homomorphic NTRU

- Level-i encryption of a is $c_a^{(i)} = [e_a/z^i]_q$
  - $|e_a| \ll q$ and $e_a = a(mod\ g)$
- To decrypt set $a \leftarrow [c_a \cdot z^i]_q\ mod\ g$
- Can add, multiply ciphertexts in R_q
  - $\left[c_a^{(i)} + c_b^{(i)}\right]_q = [(e_a + e_b)/z^i]_q = c_{a+b}^{(i)}$
    - Because $|e_a + e_b| \ll q$ and $e_a + e_b = a + b\ (mod\ g)$
  - $\left[c_a^{(i)} \cdot c_b^{(j)}\right]_q = [e_a e_b/z^{i+j}]_q = c_{ab}^{(i+j)}$
    - Because $|e_a e_b| \ll q$ and $e_a e_b = ab\ (mod\ g)$
- as long as numerator remains $\ll q$

The Public Key

- Let $f_0 = c_0^{(1)} = \frac{\alpha g}{z}$, $f_1 = c_1^{(1)} = \frac{\beta g+1}{z}$
  - $|\alpha g|, |\beta g + 1| \ll q$
- To encrypt a small m, choose small r, set $\boldsymbol{c_m^{(1)} = r f_0 + m f_1 = \frac{(r\alpha + m\beta)g + m}{z}}$
- If m is Gaussian with suitable parameter then $|m| \ll q$ and m is ~uniform mod g
  - So we can encrypt random R_g elements
  - But not any pre-set element

Zero-Test Parameter

- Need to publish information to help recognize elements of the form $\boldsymbol{rg/z^k}$
  - But not of the form $\boldsymbol{(rg+x)/z^k}$
  - Also not of the form $\boldsymbol{rg/z^{k'}}$ for $k' > k$
- First idea: publish $\boldsymbol{p_{\mathrm{zt}} = z^k/g}$
  - $[p_{zt} \cdot rg/z^k]_q = r$, with $|r| \ll q$
  - But $[p_{zt} \cdot (rg+x)/z^k]_q = [r + x/g]_q$, and x/g entails wraparound mod q
    - So typically $|[r+x/g]_q| \approx q$

Zero-Test Parameter (2)

- Main problem is that z^k/g enables also zero-testing at levels $> k$
  - E.g., $\left[\frac{rg}{z^{2k-1}} \cdot f_0 \cdot \left(\frac{z^k}{g}\right)^2\right]_q = r \cdot \alpha, \quad |r \cdot \alpha| \ll q$
- To counter this, set $\boldsymbol{p_{\mathrm{zt}} = h \cdot z^k / g}$
  - With $|h| \approx \sqrt{q}$
  - Now squaring p_{zt} already yields wraparound
- Zero-testing procedure:
  - Check if $|[p_{zt} \cdot c]_q| < q^{3/4}$

Correctness of Zero-Testing

- If $c = rg/z^k$ encodes zero at level k then $hz^k/g \cdot rg/z^k \; = hr \pmod{q}$
  - We know that $|rg| < q^{1/8}$, since all valid encodings have small numerators
  - Hence also $|r| < q^{1/8+\epsilon}$
    - This assumes g^{-1} is small in the field of fractions
  - Since $|h| < q^{1/2+\epsilon}$ then $|hr| < q^{3/4}$
- $[hz^k/g \cdot rg/z^k]_q = hr$ and $|hr| < q^{3/4}$ so the zero-test pass

Correctness of Zero-Testing (2)

- The converse is a bit more complicated:
- Let g, h be such that the two ideals gR, hR are co-prime

<u>Lemma</u>: Let e be s.t. $|eh| < q/2$ and let $w = [eh/g]_q$. If w is small enough, $|wg| < q/2$, then $e \in gR$

- i.e., $e = gr$ for some r

<u>Proof</u>: $wg = eh$ over R (since both $< q/2$) and since h, g co-prime then $g|e$.

Correctness of Zero-Testing (3)

Lemma: Let e be s.t. $|eh| < q/2$ and let $w = [eh/g]_q$. If w is small enough, $|wg| < q/2$, then $e \in gR$

Corollary: if e/z^k is a valid level-k encoding ($\rightarrow |eh| < q/2$) and it passes zero-test ($\rightarrow w$ is small), so it is an encoding of zero

Security of Zero-Testing

- This Zero-Test procedure provides functionality, not security
  - Easy to come up with an "invalid encoding" that passes the zero test.
- If we need security, publish many p_{zt}'s for many different mid-size h'es
  - Check $|[p_{zt} \cdot c]_q| < q^{3/4}$ for all of them
  - Can prove that whp over the h'es, only valid zero-encodings pass this test.

What's Hard

- Some degree-$k+1$ functions seem hard to compute, or even test

<u>Multilinear-DDH (MDDH)</u>

- For $k+1$ level-1 encoding of random elements, $c_{a_0}^{(1)}, \ldots, c_{a_k}^{(1)}$,
- and another level-k encoding $c_b^{(k)}$,
- hard to distinguish $b = a_0 \cdot \ldots \cdot a_k \ (mod\ g)$ from random b

What's Not Hard

- Other degree-$k + 1$ functions are easy

<u>Multilinear-DDH'</u>

- For $k + 1$ level-1 encoding of random elements, $c_{a_0}^{(1)}, \ldots, c_{a_k}^{(1)}$,
- and another <u>level-1</u> encoding $c_b^{(1)}$,
- easy to decide if $b = a_0 \cdot \ldots \cdot a_k \ (mod\ g)$

What's the Difference?

- A "target group" problem includes some elements encoded at the highest level (k)
  - Such problems are seemingly hard in these encodings

- A "source group" problem includes only elements encoded at levels $\leq k$
  - Include things like decision-linear assumption
  - These problems are easy, assuming that we indeed provide the public-key elements f_0, f_1

Why the Difference?

- These encodings are subject to a "weak discrete-logarithm" attacks. Given:
  - Level-i encoding of some a, and
  - Level-j encoding of 0 (e.g., f_0), with $i + j \leq k$
- Can compute "in the clear" $a' \in a + gR$
  - I.e., $a' = a + gr$ for some r
- a' is not small, so you cannot re-encode it at level 1 and break MDDH or similar
  - But if you have $g', a'_0, \ldots, a'_k$ and b', you can check whether $b' = a'_0 \cdot \ldots \cdot a'_k \bmod g'$

Dealing with "Weak DL" Attacks

- Some applications only rely on "target group" assumptions
  - Those are not affected by the attack
- More applications can get by without providing f_0, f_1, so attack does not apply
- Or use other MMAPs
  - [CTL'13] seemingly not susceptible to weak-DL
  - Can perhaps "immunize" [GGH'13] against it
    - Using GGH-encoded matrices and their eigenvalues

Computation Problems

- The source/target distinction is about decision problems
- Computation problems have their own issues
- Roughly speaking, anything that requires division is hard
  - But division in the ring R_q is easy: from $c_{a_1}^{(i)}, c_{a_2}^{(j)}$ we can compute $d = \left[c_{a_1}^{(i)} / c_{a_2}^{(j)}\right]_q$
  - d is unlikely to be a valid encoding, can perhaps be discarded using the "secure zero-test"

APPLICATIONS OF MMAPS

Application I: $(k+1)$-partite key exchange

- Public parameters include f_0, f_1, p_{zt}
- P_i draws small m_i, r_i, publishes the level-1 encoding $u_i = c_{m_i}^{(1)} = r_i f_0 + m_i f_1$
- P_i computes level-k encoding of product
$$s_i = m_i \cdot \prod_{j \neq i} u_j$$
- All parties have level-k encodings of the same thing
  - Indistinguishable from encoding of a random element, under MDDH
- How to get a shared secret key out of it?

Extracting Canonical Representation

- All of $s_0, \ldots, s_k$ encode the same thing
  - ➔ $[p_{zt}s_i - p_{zt}s_j]_q = [p_{zt}(s_i - s_j)]_q$ is small $\forall i,j$
  - ➔Roughly use MSBs of $[p_{zt} \cdot s_i]_q$ as a shared key
- Public params also include
  - Seed σ of strong randomness extractor
  - Random element $\delta \in \mathrm{R}_q$
- Shared key computed as

$$K_i = ext_\sigma\left(MSB[\delta + p_{zt} \cdot s_i]_q\right)$$

  - Whp over δ, all K_i's are equal
  - Indistinguishable to observer from random bits

Application II: Witness Encryption

- "Encryption without any key"
  - Relative to an arbitrary riddle
  - Defined here relative to exact-cover (XC)
    - Use NP-hardness to get any NP statement
- Message encrypted wrt to XC instance
  - Encryptor need not know a solution, or even if a solution exists
- Anyone with a solution can decrypt
- Semantic-security if no solution exists

Recall Exact Cover

- Instance: A universe $[n]$ and a collection of subsets $S_i \subset [n], i = 1, \dots, m$
- A solution: sub-collection of the S_i's that forms a partition of $[n]$, i.e.,
  - Subsets are pairwise disjoint, and
  - Their union is the entire $[n]$.

The [GGSW13] Construction

- On an XC instance $(n, S_1, \dots, S_m)$ and a message M
  - Use n-linear maps
  - Choose n random elements $a_1, \dots, a_n$
  - For every subset $S_i = \{j_1, \dots, j_t\}$, publish a level-$t$ encoding $c_{A_i}^{(t)}$ of $A_i = a_{j_1} \cdot \dots \cdot a_{j_t}$
  - Use a level-n encoding $c_U^{(n)}$ of $U = a_1 \cdot \dots \cdot a_n$ to encrypt, by publishing the ciphertext $C = ext_\sigma\left(MSB\left[\delta + p_{zt} \cdot c_U^{(n)}\right]_q\right) \oplus M$

The [GGSW] Construction (2)

- If $S_{i_1}, \ldots, S_{i_k}$ is a solution, then multiplying the corresponding $c_{A_i}^{(t_i)}$'s we get a level-n encoding of *U*, then we can decrypt
- Every non-solvable instance defines a computational problem
  - Distinguish a level-n encoding of U from a level-n encoding of random
- We assume all these problems to be hard
  - Is this a reasonable assumption to make?

Application III: Full-Domain Hash

- Consider the following hash function, $H : \{0,1\}^{\ell} \to$ level-ℓ-encodings:
  - Public version of Naor-Reingold PRF
  - Let $a_{1,0}, a_{1,1},\ a_{2,0}, a_{2,1},\ \ldots, a_{\ell,0}, a_{\ell,1}$ be random elements, and publish their level-1 encoding $\vec{\boldsymbol{c}} = \{c^{(1)}_{a_{i,b}} : i = 1, \ldots, \ell,\ b = 0,1\}$,

  $$H_{\vec{c}}(X) = c^{(1)}_{a_{1,X_1}} \boxtimes c^{(1)}_{a_{2,X_2}} \ldots \boxtimes c^{(1)}_{a_{\ell,X_\ell}}$$
- What can you do with it?

BLS-type Signatures [HWS13]

- Use $k = \ell + 1$, publish also $c_{a_o}^{(1)}$
  - a_0 is the secret key
- $\sigma = Sig(X) = a_0 \times H_{\vec{c}}(X)$
  - Level-ℓ encoding of an $(\ell + 1)$-product
- Verify using zero-test:

$$(\sigma \boxtimes f_1) =?= \left(c_{a_o}^{(1)} \boxtimes H_{\vec{c}}(X)\right)$$

- Can be aggregated, made identity-based

"Programmable" Hash Functions [FHPS13]

- For any ***fixed*** *"basis"* $b_1, \ldots, b_k, b^*$ (encoded at level 1), can generate a ***random*** $\vec{c}$ as above with a trapdoor td s.t.:
  - Using td we can find for any X a "representation of $H_{\vec{c}}(X)$ in this basis"
    - $H_{\vec{c}}(X) = \alpha_X \boxtimes (b_1 \boxtimes \cdots \boxtimes b_k) + B_X \boxtimes b^*$
    - α at level zero, B at level $k-1$
  - Roughly, for all but a random $1/poly$ fraction of the X'es, we have $\alpha_X = 0$
- This is useful for "partition-type" proofs of security

Obfuscation [GGHRSW13]

- Goal: take an arbitrary circuit and “encrypt it”, so that:
  - Can still evaluate the result on any input
  - But “not much else”
- Formulating “not much else” is hard
  - [BGIRSVY01] show that some natural formulations cannot be met
  - Also defined the weaker notion of “indistinguishability Obfuscation” (iO):
  - If C_1, C_2 compute the same function, then $OBF(C_1) \approx OBF(C_2)$

iO for NC^1

- Begin with a corollary of Barrington's theorem, we can recognize $\mathrm{L} \in NC^1$ via matrix multiplication:
    - C_L represented by a sequence of matrices of length $\exp\left(depth(C_L)\right)$
    - Input x determines a sub-sequence
    - $x \in L$ iff their product is the identity

Obfuscating C_L

- Randomize the matrices for C_L
  - How to randomize is the hard part, need to counter several attacks
- Provide level-1 encoding of matrices
- To evaluate on x
  - Choose a subset and multiply the encoding
  - Use zero-testing to check for identity

Security

- Mostly heuristic, but supported by generic-group arguments
- Every pair of circuits C_1, C_2, defines a decision problem
  - We assume that they are all hard
- These are all source-group assumptions
  - Since the matrices are encoded at level 1
  - But we are not giving f_0, f_1, so the weak-DL attack does not apply

Summary

- Can approximate cryptographic MMAPs
  - Using SWHE with "handicapped secret key"
  - Known constructions from NTRU, "integer HE"
  - Can we do the same thing from other schemes?
- Enabling many new applications
  - But hardness assumptions are strong, "ugly"
  - In desperate need of a coherent theory
- Practical performance lacking
  - Worse than the [Gen09] HE scheme