CS 65500
Advanced Cryptography

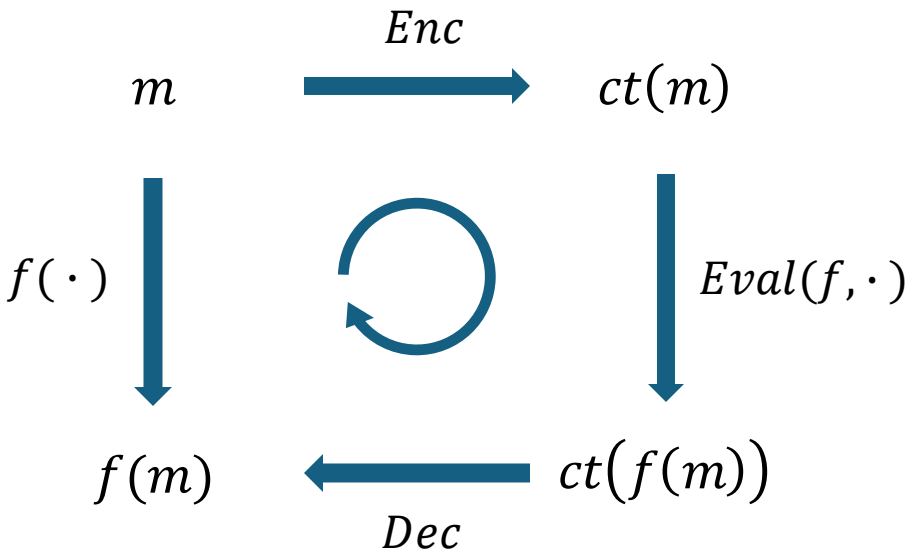
Guest Lecture: Fully Homomorphic Encryption

Keewoo Lee
UC Berkeley

Fully Homomorphic Encryption (FHE)

- FHE is a cryptosystem that supports arbitrary computation on encrypted data
- $FHE = (KeyGen, Enc, Dec, \boldsymbol{Eval})$
  - $sk \leftarrow KeyGen(1^{\lambda})$
  - $ct(m) \leftarrow Enc(sk, m)$
  - $m \leftarrow Dec(sk, ct(m))$
  - $ct(f(m)) \leftarrow Eval(f, ct(m))$

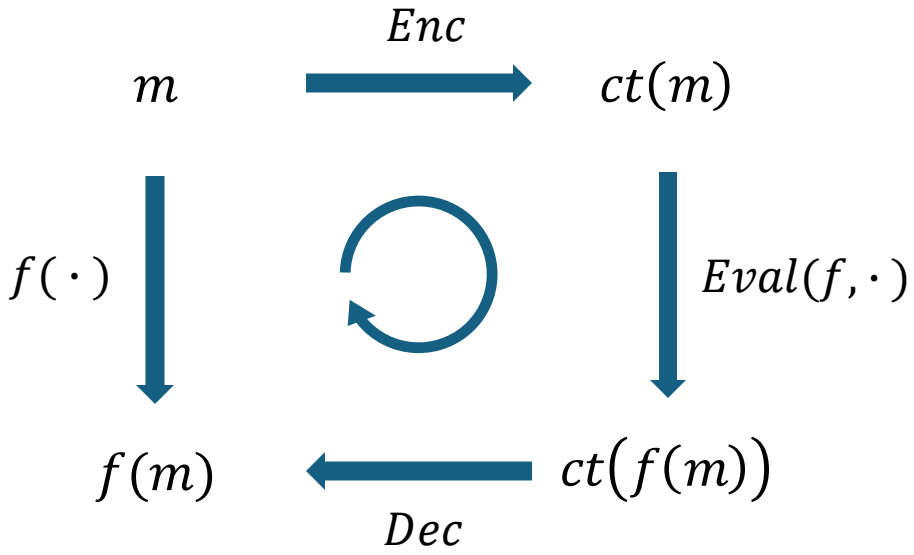
sk-FHE; Will come back to this later.



- We require Correctness & Semantic Security
- Also “Compactness”
  - The output length of $Eval(f, \cdot)$ should depend only on the output length of $f(\cdot)$.
  - Otherwise, FHE is trivial to achieve. (why?)

Applications of FHE

- Privacy-preserving Outsourced Computation
  - Secure Cloud Computing
  - End-to-End Encrypted Web Services
  - Direct construction of PIR



- Round-optimal & Communication-efficient 2PC
  - (with some caveats: semi-honest & circuit privacy)
  - Direct construction of (semi-honest) PSI

Meme Credit: Leo Ducas

Brief History of FHE

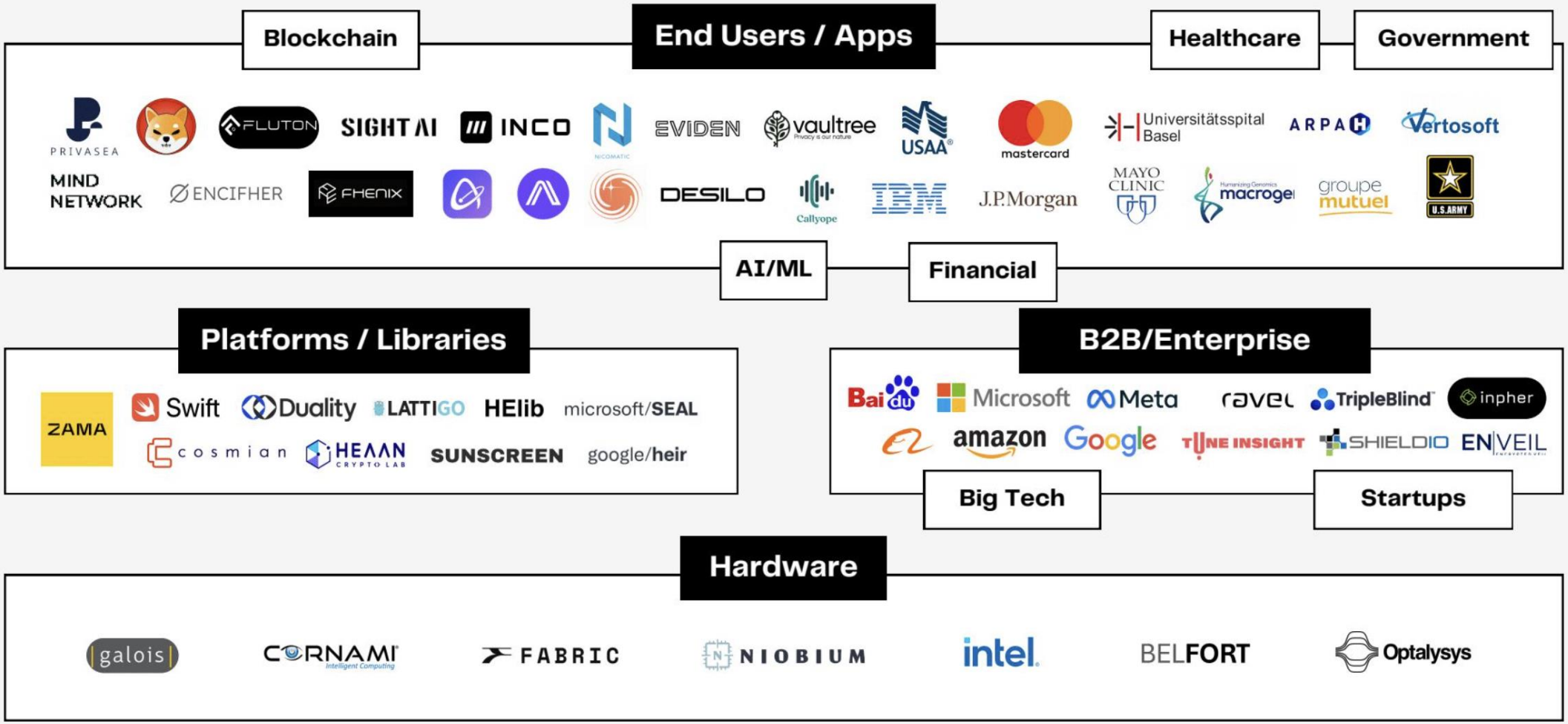
- (1978) Problem Formulation [Rivest-Adleman-Dertouzos;78]
  - c.f. RSA [Rivest-Shamir-Adleman;77]
- 30 Years of Dark Age
  - Partial Solutions & Negative Results
  - "Holy Grail" of Cryptography

Q. Why is it so challenging and intriguing?

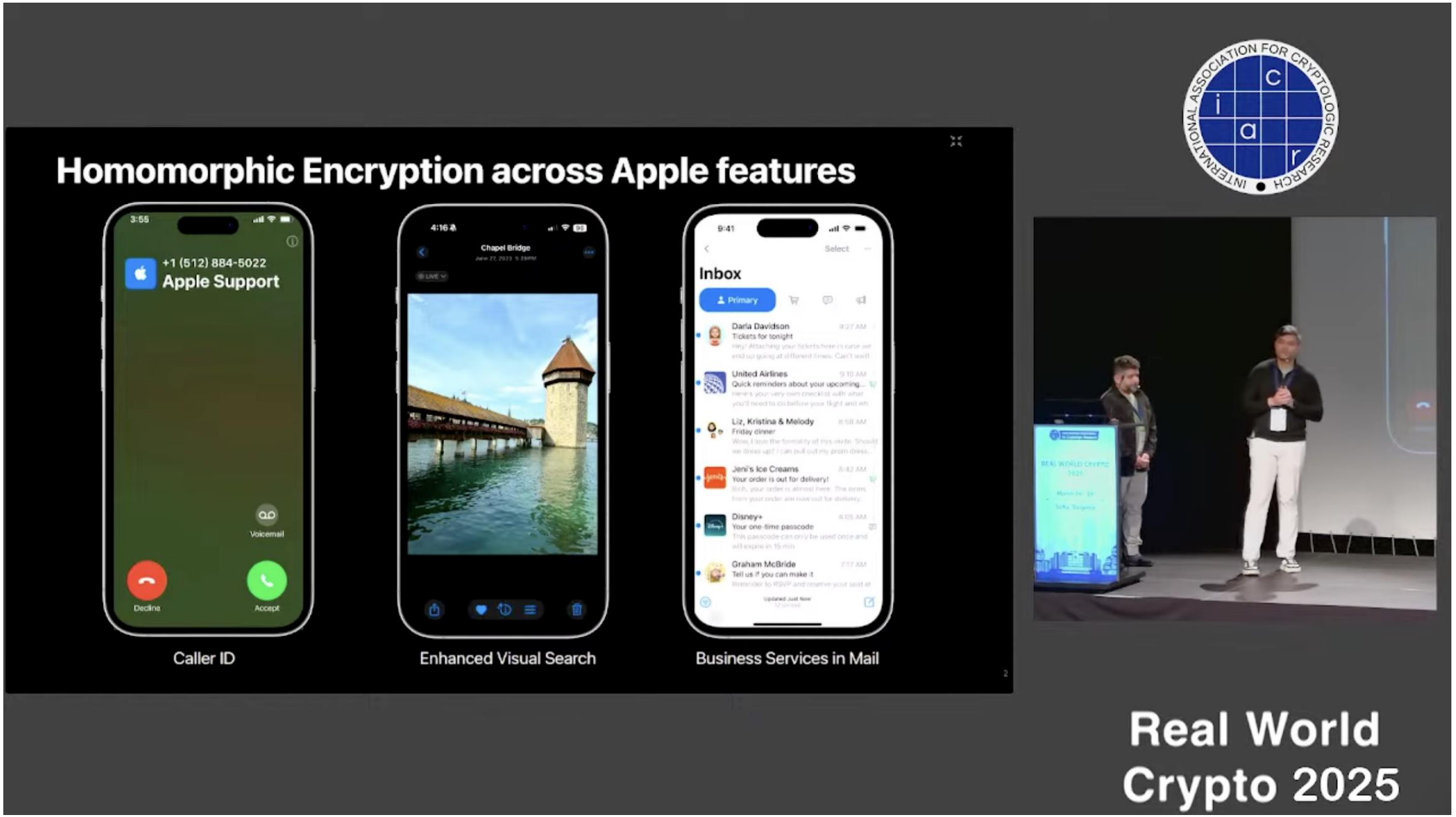
The tension between **structure** and **hardness**, and between **functionality** and **privacy**.

- (2009) Breakthrough: First FHE Construction [Gentry;STOC'09]
  - Gödel Prize 2022: Brakerski-Gentry-Vaikuntanathan
- 15 Years of Intensive Research
  - Rapid improvement from theory to practice (outpacing the Moore's law!)
    - ✓ Boneh@CCS'24: "FHE will be used for everything(?)"

THE STATE OF FHE Market Map*



*Is your FHE project missing? Get in touch at zama.ai/the-state-of-fhe-report



Apple's Real World Deployment of Homomorphic Encryption at Scale
https://www.youtube.com/live/R1NEfuv3iMk?t=16963s

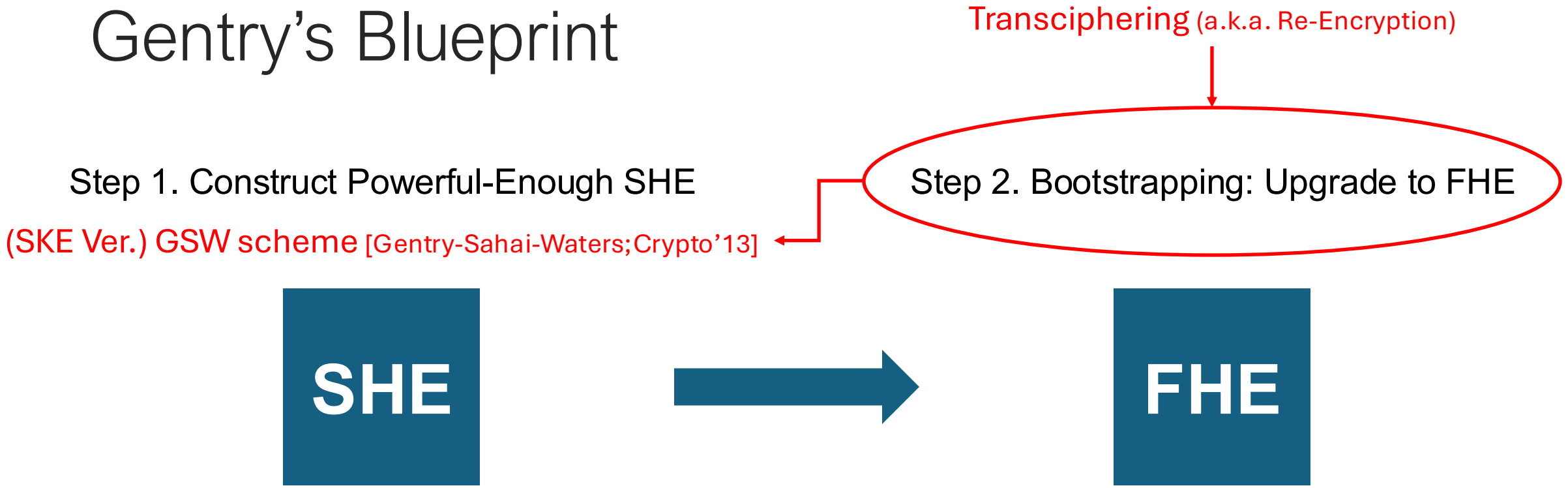
The Egyptian god protocols: three powerful and highly general-purpose constructions that can let us do computation on data while at the same time keeping the data completely private.

https://vitalik.eth.limo/general/2025/04/14/privacy.html

Taxonomy of HE

- Disclaimer: These terminologies are not universally agreed upon
- Fully Homomorphic Encryption (FHE)
  - A single parameter set can support arbitrary circuits.
- Somewhat Homomorphic Encryption (SHE)
  - For any circuit, we can choose a parameter set to support it.
  - e.g. Can support circuits only up to a certain depth (a.k.a. Leveled FHE)
- Partially Homomorphic Encryption
  - Fundamental restriction on supported homomorphic circuits
  - e.g. Group HE (Additive HE, Multiplicative HE), Linearly HE, etc

Gentry's Blueprint



- Enough to evaluate ***its own decryption circuit,*** plus a small additional margin. (The Key Idea of Bootstrapping)
- Challenge: Greater depth requires larger parameters, which in turn necessitate even greater depth, and so on.
- Prior to [Gentry;STOC'09], no such SHE were known.

Transciphering (a.k.a. Re-encryption) [Lauter-Naehrig-Vaikuntanathan;11]

- Goal: Securely delegate the ability to convert SKE (e.g. AES) ctxt into FHE ctxt
  - Naively providing sk_{SKE} compromises privacy
- Motivation: Expansion Ratio of FHE is Large
  - Reduce communication and server storage by sending and storing SKE ctxt.
  - Server converts to FHE ctxt only at use.
- Idea: **Homomorphic Decryption!**

Transciphering (a.k.a. Re-encryption) [Lauter-Naehrig-Vaikuntanathan;11]

- Publish $ct_{FHE}(sk_{SKE})$ as transciphering key
- To transcipher $ct_{SKE}(m)$, compute the following (Homomorphic Decryption):

$$Eval_{FHE}\left(Dec_{SKE}(\ \cdot\ , ct_{SKE}(m)), ct_{FHE}(sk_{SKE})\right)$$

$$= ct_{FHE}\left(Dec_{SKE}(sk_{SKE}, ct_{SKE}(m))\right)$$

$$= ct_{FHE}(m)$$

$Eval_{FHE}(f, ct_{FHE}(x))$
$= ct_{FHE}(f(x))$

$Dec_{SKE}(sk_{SKE}, ct_{SKE}(x))$
$= x$

Transciphering to SHE

- Publish $ct_{SHE}(sk_{SKE})$ as transciphering key
- To transcipher $ct_{SKE}(m)$, compute the following (Homomorphic Decryption):

$$Eval_{SHE}\left(Dec_{SKE}(\,\cdot\,, ct_{SKE}(m)), ct_{SHE}(sk_{SKE})\right)$$

$$= ct_{SHE}\left(Dec_{SKE}(sk_{SKE}, ct_{SKE}(m))\right)$$

$$= ct_{SHE}(m)$$

$$Eval_{SHE}(f, ct_{SHE}(x)) = ct_{SHE}(f(x))$$

- Holds if SHE supports $f = Dec_{SKE}(\,\cdot\,, ct_{SKE}(m))$, assuming the transciphering key is a fresh ctxt.

Transciphering to SHE

- Publish $ct_{SHE}(sk_{SKE})$ as transciphering key
- To transcipher $ct_{SKE}(m)$, compute the following (Homomorphic Decryption):

$$Eval_{SHE}\left(F \circ Dec_{SKE}(\,\cdot\,, ct_{SKE}(m)), ct_{SHE}(sk_{SKE})\right)$$

$$= ct_{SHE}\left(F \circ Dec_{SKE}(sk_{SKE}, ct_{SKE}(m))\right)$$

$$= ct_{SHE}(F(m))$$

$$Eval_{SHE}(f, ct_{SHE}(x)) = ct_{SHE}(f(x))$$

- Holds if SHE supports $f = F \circ Dec_{SKE}(\,\cdot\,, ct_{SKE}(m))$, assuming the transciphering key is a fresh ctxt.

Bootstrapping: SHE Self-Transciphering

- Publish $ct_{SHE}(sk_{SHE})$ as bootstrapping key
- To compute F on $ct_{SHE}(m)$, compute the following (Homomorphic Decryption):

$$Eval_{SHE}\left(F \circ Dec_{SHE}(\,\cdot\,, ct_{SHE}(m)), ct_{SHE}(sk_{SHE})\right)$$

$$= ct_{SHE}\left(F \circ Dec_{SHE}(sk_{SHE}, ct_{SHE}(m))\right)$$

$$= ct_{SHE}(F(m))$$

$$Eval_{SHE}(f, ct_{SHE}(x)) = ct_{SHE}(f(x))$$

- Holds if SHE supports $f = F \circ Dec_{SHE}(\,\cdot\,, ct_{SHE}(m))$, assuming the bootstrapping key is a fresh ctxt.

This procedure requires no homomorphic property of $ct_{SHE}(m)$ other than its decryptability!!!

Bootstrapping: SHE Self-Transciphering

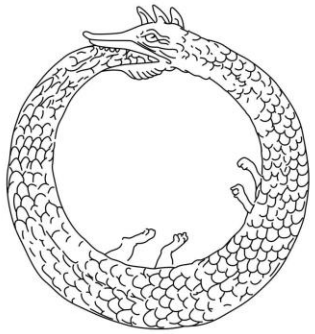
- Publish $ct_{SHE}(sk_{SHE})$ as bootstrapping key
- To compute F on $ct_{SHE}(m)$, compute the following (Homomorphic Decryption):

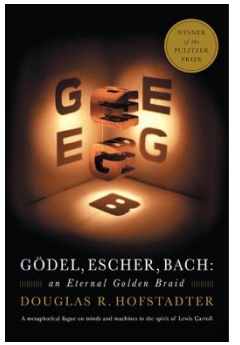
$$Eval_{SHE}\left(F \circ Dec_{SHE}(\ \cdot\ , ct_{SHE}(m)), ct_{SHE}(sk_{SHE})\right)$$

Theorem. Let $SHE = (KeyGen, Enc, Dec, Eval)$ be a "secure" SHE scheme that supports homomorphic evaluation of $f(x) = Dec(x, ct_1)\ NAND\ Dec(x, ct_2)$.

Then, SHE can be turned into am FHE scheme.

Bootstrapping





- Bootstrapping SHE is homomorphically evaluating its own decryption circuit.
- Gentry's Bootstrapping is interesting in many ways.
  - Self-reference
  - Connection to Circular Security

Circular Security

- IND-CPA security says nothing about circular security: We can convert any IND-CPA secure encryption scheme into one that remains IND-CPA secure but is circularly insecure. (How?)
- Constructing a provably circularly secure bootstrappable SHE remains a major open problem.
- Nonetheless, we believe that natural SHE schemes, including GSW, do satisfy circular security.

Q. Are we allowed to do this?

A. Not in general…

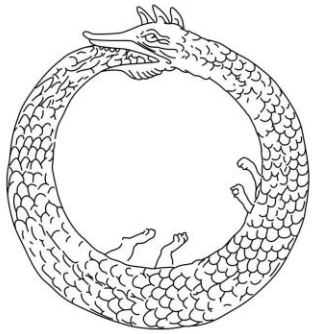
- Publish $ct_{SHE}(sk_{SHE})$ as bootstrapping key

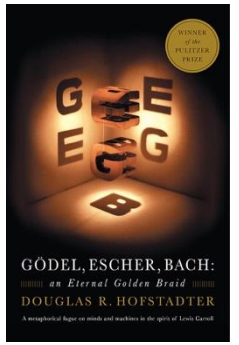
a circularly secure SHE scheme

Theorem. Let $SHE = (KeyGen, Enc, Dec, Eval)$ be a "secure" SHE scheme that supports homomorphic evaluation of $f(x) = Dec(x, ct_1)\ NAND\ Dec(x, ct_2)$.

Then, SHE can be turned into am FHE scheme.

Bootstrapping





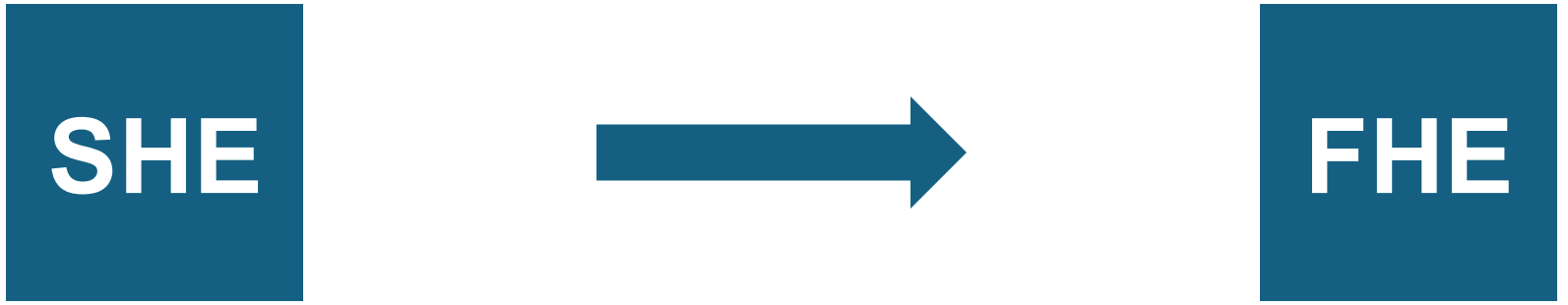
- Bootstrapping SHE is homomorphically evaluating its own decryption circuit.
- Gentry's Bootstrapping is interesting in many ways.
  - Self-reference
  - Connection to Circular Security
  - Non-blackbox technique
- Gentry's Real Thought Process:
  - "What would be the coolest circuit for an HE scheme to evaluate?"
  - "Probably its own decryption circuit."
  - "Eureka!"

Gentry's Blueprint

Step 1. Construct Powerful-Enough SHE

(SKE Ver.) GSW scheme [Gentry-Sahai-Waters;Crypto'13]

Step 2. Bootstrapping: Upgrade to FHE



- Enough to evaluate ***its own decryption circuit,*** plus a small additional margin. (The Key Idea of Bootstrapping)
- Challenge: Greater depth requires larger parameters, which in turn necessitate even greater depth, and so on.
- Prior to [Gentry;STOC'09], no such SHE were known.

Why GSW over other schemes?

- Disclaimer: Vanilla GSW is rarely used in “practice”
  - Expansion Ratio $> 2^{30}$: 1-bit message → at least 1Gbit ctxt
  - c.f. Expansion Ratio of AES (resp. ElGamal) is 1 (resp. 2)
- Easy-to-understand
  - AGCD-based vDGHV is also fairly intuitive but is no longer considered mainstream
- Foundation of other constructions
  - A ring variant of GSW serves as a building block of the TFHE scheme
    - ✓ TFHE, supported by Zama, is currently one of the leading schemes (c.f. CKKS, BGV/BFV)
  - Main technique used in GSW is extremely versatile across various other primitives
    - ✓ Attribute-Based Encryption, Homomorphic Signature, Constrained PRF ...

Why sk-FHE instead of pk-FHE?

- sk-FHE already enables interesting applications
  - In fact, all the applications mentioned so far require only sk-FHE!
    - ✓ There are three players in FHE application scenarios: encryptor, evaluator, decryptor.
    - ✓ In many cases, encryptor = decryptor.
- We have a generic conversion from sk-FHE to pk-FHE [Rothblum;TCC'11]
  - Idea: Publish a bunch of encryptions of zero as the public key.
    - ✓ As the final step of encryption, homomorphically add a random subset of them.
    - ✓ Security follows from Leftover Hash Lemma
    - ✓ In the same spirit as Regev's LWE-based PKE [Regev;STOC'05]
  - Interesting in the sense that Minicrypt+Homomorphism implies Cryptomania
    - ✓ c.f. black-box separation between Minicrypt and Cryptomania

Appendix

Limitations of Vanilla FHE

- Circuit Privacy
  - Does the evaluated ctxt leak any information about the circuit applied to it?
  - c.f. Noise-flooding [Gentry;STOC'09], Sanitization [Ducas-Stehlé;Eurocrypt'16]
- Key Management
  - In MPC protocols with more than two parties, who should hold the secret key?
  - c.f. Threshold FHE & Multi-key FHE
- Verifiability
  - For malicious security, we must ensure the evaluator did their job correctly.
  - c.f. Verifiable FHE

Limitations of Vanilla FHE (cont.)

- Computational Model
  - Modern FHE schemes are based on the Circuit Model
    - ✓ in contrast to the RAM Model (w/ IF-THEN-ELSE)
    - ✓ c.f. Binary Search, Quicksort
    - ✓ Challenge: By definition, runtime and output length cannot depend on the input
  - Certain applications align well with the circuit model
    - ✓ e.g. ML Inference
  - But other applications require some care
    - ✓ e.g. ML Training
  - c.f. RAM-FHE [Lin-Mook-Wichs;STOC'23]