Fully Homomorphic Encryption

Part I

Mehdi Tibouchi

NTT Secure Platform Laboratories

EPIT 2013, 2013–03–21

incl. some slides courtesy of J.-S. Coron

Outline

Homomorphic encryption

- Computing on encrypted data.
- Multiplicatively homomorphic: "textbook RSA".

$$\left.\begin{aligned} c_1 &= m_1^e \bmod N \\ c_2 &= m_2^e \bmod N \end{aligned}\right. \Rightarrow c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod N$$

- Additively homomorphic: Paillier.

$$\left.\begin{aligned} c_1 &= g^{m_1} x_1^N \bmod N^2 \\ c_2 &= g^{m_2} x_2^N \bmod N^2 \end{aligned}\right. \Rightarrow c_1 \cdot c_2 = g^{m_1+m_2\ [N]} (x_1 x_2)^N \bmod N^2$$

- Fully homomorphic: homomorphic for both addition and multiplication
  - Open problem until Gentry's breakthrough in 2009.

Homomorphic encryption

- Computing on encrypted data.
- Multiplicatively homomorphic: "textbook RSA".

$$\left.\begin{aligned} c_1 &= m_1^e \bmod N \\ c_2 &= m_2^e \bmod N \end{aligned}\right. \Rightarrow c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod N$$

- Additively homomorphic: Paillier.

$$\left.\begin{aligned} c_1 &= g^{m_1} x_1^N \bmod N^2 \\ c_2 &= g^{m_2} x_2^N \bmod N^2 \end{aligned}\right. \Rightarrow c_1 \cdot c_2 = g^{m_1 + m_2\ [N]} (x_1 x_2)^N \bmod N^2$$

- Fully homomorphic: homomorphic for both addition and multiplication
  - Open problem until Gentry's breakthrough in 2009.

Homomorphic encryption

- Computing on encrypted data.
- Multiplicatively homomorphic: "textbook RSA".

$$\begin{aligned} c_1 &= m_1^e \bmod N \\ c_2 &= m_2^e \bmod N \end{aligned} \Rightarrow c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod N$$

- Additively homomorphic: Paillier.

$$\begin{aligned} c_1 &= g^{m_1} x_1^N \bmod N^2 \\ c_2 &= g^{m_2} x_2^N \bmod N^2 \end{aligned} \Rightarrow c_1 \cdot c_2 = g^{m_1 + m_2 \, [N]} (x_1 x_2)^N \bmod N^2$$

- Fully homomorphic: homomorphic for both addition and multiplication
  - Open problem until Gentry's breakthrough in 2009.

Homomorphic encryption

- Computing on encrypted data.
- Multiplicatively homomorphic: “textbook RSA”.

$$\begin{aligned} c_1 &= m_1^e \bmod N \\ c_2 &= m_2^e \bmod N \end{aligned} \Rightarrow c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod N$$

- Additively homomorphic: Paillier.

$$\begin{aligned} c_1 &= g^{m_1} x_1^N \bmod N^2 \\ c_2 &= g^{m_2} x_2^N \bmod N^2 \end{aligned} \Rightarrow c_1 \cdot c_2 = g^{m_1 + m_2\ [N]} (x_1 x_2)^N \bmod N^2$$

- Fully homomorphic: homomorphic for both addition and multiplication
  - Open problem until Gentry’s breakthrough in 2009.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Homomorphic encryption for e-voting

- A typical application of additively homomorphic encryption is secure voting schemes.
- In a yes-no election, each voter casts a ballot by encrypting 0 or 1 using the Paillier public key of the organizer of the election.
- The ballots are then shuffled and added together homomorphically by some independent third parties.
- Decrypting the resulting ciphertext reveals the tally, while individual votes remain secret.
- Add in zero-knowledge proofs to ensure that each step is correct, and use threshold encryption/secret sharing to avoid a single authority.
- Make sure voters have Ph.D.'s in cryptography to understand the whole process.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Timeline of privacy homomorphisms

- [RSA77]: multiplication mod N;
- [RAD78]: introduce the notion of privacy homomorphism
  - almost suggests FHE as an open problem...
- [GM84]: addition mod 2, CPA-security;
- [ElGamal84]: multiplication mod p;
- [Paillier98], [OU98]: addition mod N (resp. mod p);
- [BGN06]: polynomials of degree 2 mod p;
- [Gentry09]: addition and multiplication mod 2!
  - a.k.a. fully homomorphic encryption.

Fully homomorphic encryption

- We restrict ourselves to encrypting a single bit:
  - $0 \to 203\text{ef}6124\ldots23\text{ab}87_{16}$
  - $1 \to \text{b}327653\text{c}1\ldots\text{db}3265_{16}$
  - no loss of generality, by the hybrid argument.
- Fully homomorphic property
  - Given $E(b_0)$ and $E(b_1)$, one can compute $E(b_0 \oplus b_1)$ and $E(b_0 \cdot b_1)$ without knowing the private key.
- Computing over a ring:
  - Given a circuit with xors and ands, and encrypted input bits, one can compute the output in encrypted form, without knowing the private key.
  - Hence, compute any function on encrypted data that can be represented as a boolean circuit with polynomially many gates (and $\mathbf{BPP} \subseteq \mathbf{P/poly}$).

Fully homomorphic encryption

- We restrict ourselves to encrypting a single bit:
  - $0 \to 203\text{ef}6124\ldots23\text{ab}87_{16}$
  - $1 \to \text{b}327653\text{c}1\ldots\text{db}3265_{16}$
  - no loss of generality, by the hybrid argument.
- Fully homomorphic property
  - Given $E(b_0)$ and $E(b_1)$, one can compute $E(b_0 \oplus b_1)$ and $E(b_0 \cdot b_1)$ without knowing the private key.
- Computing over a ring:
  - Given a circuit with xors and ands, and encrypted input bits, one can compute the output in encrypted form, without knowing the private key.
  - Hence, compute any function on encrypted data that can be represented as a boolean circuit with polynomially many gates (and $\mathbf{BPP} \subseteq \mathbf{P/poly}$).

Fully homomorphic encryption

- We restrict ourselves to encrypting a single bit:
  - $0 \to 203\text{ef}6124 \ldots 23\text{ab}87_{16}$
  - $1 \to \text{b}327653\text{c}1 \ldots \text{db}3265_{16}$
  - no loss of generality, by the hybrid argument.
- Fully homomorphic property
  - Given $E(b_0)$ and $E(b_1)$, one can compute $E(b_0 \oplus b_1)$ and $E(b_0 \cdot b_1)$ without knowing the private key.
- Computing over a ring:
  - Given a circuit with xors and ands, and encrypted input bits, one can compute the output in encrypted form, without knowing the private key.
  - Hence, compute any function on encrypted data that can be represented as a boolean circuit with polynomially many gates (and **BPP** $\subseteq$ **P/poly**).

Security notions

- One can consider both secret-key and public-key FHE schemes.
  - Both are interesting.
- Usual security notion: IND-CPA.
  - In the view of an adversary without the secret/private key, $E(0) \cong E(1)$.
- Can we have more?
  - CCA2 is incompatible with homomorphic properties.
  - CCA1 is possible, some conversions exist.

Security notions

- One can consider both secret-key and public-key FHE schemes.
  - Both are interesting.
- Usual security notion: IND-CPA.
  - In the view of an adversary without the secret/private key, $E(0) \cong E(1)$.
- Can we have more?
  - CCA2 is incompatible with homomorphic properties.
  - CCA1 is possible, some conversions exist.

Security notions

- One can consider both secret-key and public-key FHE schemes.
  - Both are interesting.
- Usual security notion: IND-CPA.
  - In the view of an adversary without the secret/private key, $E(0) \cong E(1)$.
- Can we have more?
  - CCA2 is incompatible with homomorphic properties.
  - CCA1 is possible, some conversions exist.

Compactness

- Trivial construction of FHE from any encryption scheme:
  - Same key generation and encryption;
  - $\mathsf{Eval}(\mathsf{pk}, f, c) = (c, f)$;
  - $\mathsf{Decrypt}(\mathsf{sk}, (c, f)) = f(\mathsf{Decrypt}(\mathsf{sk}, c))$.
- We want to exclude such trivial constructions, where no computation is actually carried out on ciphertexts.
- Usual extra requirement: compactness.
  - Ciphertext size independent of successive homomorphic operations.
- One can ask for something stronger: circuit privacy.
  - For a given plaintext, ciphertext distribution independent of successive homomorphic operations.
  - Rather costly to achieve; usually relaxed somewhat in "practical" schemes.

Compactness

- Trivial construction of FHE from any encryption scheme:
  - Same key generation and encryption;
  - $\mathsf{Eval}(\mathsf{pk}, f, c) = (c, f)$;
  - $\mathsf{Decrypt}(\mathsf{sk}, (c, f)) = f(\mathsf{Decrypt}(\mathsf{sk}, c))$.
- We want to exclude such trivial constructions, where no computation is actually carried out on ciphertexts.
- Usual extra requirement: compactness.
  - Ciphertext size independent of successive homomorphic operations.
- One can ask for something stronger: circuit privacy.
  - For a given plaintext, ciphertext distribution independent of successive homomorphic operations.
  - Rather costly to achieve; usually relaxed somewhat in "practical" schemes.

Compactness

- Trivial construction of FHE from any encryption scheme:
  - Same key generation and encryption;
  - $\mathsf{Eval}(\mathsf{pk}, f, c) = (c, f)$;
  - $\mathsf{Decrypt}(\mathsf{sk}, (c, f)) = f(\mathsf{Decrypt}(\mathsf{sk}, c))$.
- We want to exclude such trivial constructions, where no computation is actually carried out on ciphertexts.
- Usual extra requirement: compactness.
  - Ciphertext size independent of successive homomorphic operations.
- One can ask for something stronger: circuit privacy.
  - For a given plaintext, ciphertext distribution independent of successive homomorphic operations.
  - Rather costly to achieve; usually relaxed somewhat in "practical" schemes.

Compactness

- Trivial construction of FHE from any encryption scheme:
  - Same key generation and encryption;
  - $\mathsf{Eval}(\mathsf{pk}, f, c) = (c, f)$;
  - $\mathsf{Decrypt}(\mathsf{sk}, (c, f)) = f(\mathsf{Decrypt}(\mathsf{sk}, c))$.
- We want to exclude such trivial constructions, where no computation is actually carried out on ciphertexts.
- Usual extra requirement: compactness.
  - Ciphertext size independent of successive homomorphic operations.
- One can ask for something stronger: circuit privacy.
  - For a given plaintext, ciphertext distribution independent of successive homomorphic operations.
  - Rather costly to achieve; usually relaxed somewhat in "practical" schemes.

Outline

What can we do with FHE? (1)

- Recall the secure voting protocol from a few slides back:
  - To cast their ballots, voters encrypt $x_i = 0$ or 1 under an additive homomorphic encryption scheme, together with a zero-knowledge proof of equality to 0 or 1;
  - Third parties shuffle the ballots, add the ciphertexts homomorphically, checking all the proofs;
  - Organizers decrypt the tally.
- Using fully homomorphic encryption, do away with the voters' zero-knowledge proofs:
  - In addition to computing the homomorphic sum of the ballots, the third parties can compute a ciphertext for:

$$t = \prod x_i(x_i - 1).$$

  - The organizers can decrypt this ciphertext and check that $t = 0$ to ensure all ballots were valid (equal to 0 or 1).
- Not really an improvement of existing voting protocols, but gives an idea of what you can do.

What can we do with FHE? (1)

- Recall the secure voting protocol from a few slides back:
  - To cast their ballots, voters encrypt $x_i = 0$ or 1 under an additive homomorphic encryption scheme, together with a zero-knowledge proof of equality to 0 or 1;
  - Third parties shuffle the ballots, add the ciphertexts homomorphically, checking all the proofs;
  - Organizers decrypt the tally.
- Using fully homomorphic encryption, do away with the voters' zero-knowledge proofs:
  - In addition to computing the homomorphic sum of the ballots, the third parties can compute a ciphertext for:

$$t = \prod x_i(x_i - 1).$$

  - The organizers can decrypt this ciphertext and check that $t = 0$ to ensure all ballots were valid (equal to 0 or 1).
- Not really an improvement of existing voting protocols, but gives an idea of what you can do.

What can we do with FHE? (1)

- Recall the secure voting protocol from a few slides back:
  - To cast their ballots, voters encrypt $x_i = 0$ or 1 under an additive homomorphic encryption scheme, together with a zero-knowledge proof of equality to 0 or 1;
  - Third parties shuffle the ballots, add the ciphertexts homomorphically, checking all the proofs;
  - Organizers decrypt the tally.
- Using fully homomorphic encryption, do away with the voters' zero-knowledge proofs:
  - In addition to computing the homomorphic sum of the ballots, the third parties can compute a ciphertext for:

$$t = \prod x_i(x_i - 1).$$

  - The organizers can decrypt this ciphertext and check that $t = 0$ to ensure all ballots were valid (equal to 0 or 1).
- Not really an improvement of existing voting protocols, but gives an idea of what you can do.

What can we do with FHE? (2)

- Possibe business model courtesy of J.-S. Coron:
- You have a software that given the revenue, past income, headcount, etc., of a company can predict its future stock price.
  - I want to know the future stock price of my company, but I don't want to disclose confidential information.
  - And you don't want to give me your software containing secret formulas.
- Using homomorphic encryption:
  - I encrypt all the inputs using fully homomorphic encryption and send them to you in encrypted form.
  - You process all my inputs, viewing your software as a circuit.
  - You send me the result, still encrypted.
  - You didn't learn any information about my company.

What can we do with FHE? (2)

- Possibe business model courtesy of J.-S. Coron:
- You have a software that given the revenue, past income, headcount, etc., of a company can predict its future stock price.
  - I want to know the future stock price of my company, but I don't want to disclose confidential information.
  - And you don't want to give me your software containing secret formulas.
- Using homomorphic encryption:
  - I encrypt all the inputs using fully homomorphic encryption and send them to you in encrypted form.
  - You process all my inputs, viewing your software as a circuit.
  - You send me the result, still encrypted.
  - You didn't learn any information about my company.

What can we do with FHE? (2)

- Possibe business model courtesy of J.-S. Coron:
- You have a software that given the revenue, past income, headcount, etc., of a company can predict its future stock price.
  - I want to know the future stock price of my company, but I don't want to disclose confidential information.
  - And you don't want to give me your software containing secret formulas.
- Using homomorphic encryption:
  - I encrypt all the inputs using fully homomorphic encryption and send them to you in encrypted form.
  - You process all my inputs, viewing your software as a circuit.
  - You send me the result, still encrypted.
  - You didn't learn any information about my company.

What can we do with FHE? (3)

- Some say FHE is a very nice solution in search of a problem.
- Applications I do not believe in:
  - Fully homomorphic Google queries.
  - Or anything about secure cloud computing, really.
  - Because cloud computing is database search, and doing this with encrypted queries is intrinsically inefficient (linear instead of logarithmic in the size of the database).
  - Unless interactive protocols are fine, but then use PIR.
- Applications that may see the light of day:
  - Handling of data sensitive enough that parties are prepared to pay a heavy price for extra security;
  - involving relatively simple functions (shallow circuits).
- In the meantime, FHE is a powerful crypto primitive that lets you build many advanced protocols (NIZK, MPC, secure databases, etc.) provided you do not care to much about efficiency.

What can we do with FHE? (3)

- Some say FHE is a very nice solution in search of a problem.
- Applications I do not believe in:
  - Fully homomorphic Google queries.
  - Or anything about secure cloud computing, really.
  - Because cloud computing is database search, and doing this with encrypted queries is intrinsically inefficient (linear instead of logarithmic in the size of the database).
  - Unless interactive protocols are fine, but then use PIR.
- Applications that may see the light of day:
  - Handling of data sensitive enough that parties are prepared to pay a heavy price for extra security;
  - involving relatively simple functions (shallow circuits).
- In the meantime, FHE is a powerful crypto primitive that lets you build many advanced protocols (NIZK, MPC, secure databases, etc.) provided you do not care to much about efficiency.

What can we do with FHE? (3)

- Some say FHE is a very nice solution in search of a problem.
- Applications I do not believe in:
  - Fully homomorphic Google queries.
  - Or anything about secure cloud computing, really.
  - Because cloud computing is database search, and doing this with encrypted queries is intrinsically inefficient (linear instead of logarithmic in the size of the database).
  - Unless interactive protocols are fine, but then use PIR.
- Applications that may see the light of day:
  - Handling of data sensitive enough that parties are prepared to pay a heavy price for extra security;
  - involving relatively simple functions (shallow circuits).
- In the meantime, FHE is a powerful crypto primitive that lets you build many advanced protocols (NIZK, MPC, secure databases, etc.) provided you do not care to much about efficiency.

What can we do with FHE? (3)

- Some say FHE is a very nice solution in search of a problem.
- Applications I do not believe in:
  - Fully homomorphic Google queries.
  - Or anything about secure cloud computing, really.
  - Because cloud computing is database search, and doing this with encrypted queries is intrinsically inefficient (linear instead of logarithmic in the size of the database).
  - Unless interactive protocols are fine, but then use PIR.
- Applications that may see the light of day:
  - Handling of data sensitive enough that parties are prepared to pay a heavy price for extra security;
  - involving relatively simple functions (shallow circuits).
- In the meantime, FHE is a powerful crypto primitive that lets you build many advanced protocols (NIZK, MPC, secure databases, etc.) provided you do not care to much about efficiency.

Efficiency is improving

- 2009: breakthrough scheme by Gentry.
  - Concrete parameters unclear, probably prohibitively inefficient.
- 2010: vDGHV scheme over the integers.
  - Public key size $> 2^{60}$ bits at reasonable security levels!
- 2011: first implementations of these schemes with numerous optimizations [GH11], [CMNT11].
  - 15–30 min. per multiplication gate, public key $\approx$ 1 GB.
- 2011–2013: (R)LWE-based schemes.
  - Simpler, more efficient, more versatile.
  - Optimized implementations evaluate the full AES circuit in 10–30 min. amortized.
  - Lauter et al. announce performance in the millisecond per gate range for shallow circuits.

Efficiency is improving

- 2009: breakthrough scheme by Gentry.
  - Concrete parameters unclear, probably prohibitively inefficient.
- 2010: vDGHV scheme over the integers.
  - Public key size $> 2^{60}$ bits at reasonable security levels!
- 2011: first implementations of these schemes with numerous optimizations [GH11], [CMNT11].
  - 15–30 min. per multiplication gate, public key $\approx$ 1 GB.
- 2011–2013: (R)LWE-based schemes.
  - Simpler, more efficient, more versatile.
  - Optimized implementations evaluate the full AES circuit in 10–30 min. amortized.
  - Lauter et al. announce performance in the millisecond per gate range for shallow circuits.

Efficiency is improving

- 2009: breakthrough scheme by Gentry.
  - Concrete parameters unclear, probably prohibitively inefficient.
- 2010: vDGHV scheme over the integers.
  - Public key size $> 2^{60}$ bits at reasonable security levels!
- 2011: first implementations of these schemes with numerous optimizations [GH11], [CMNT11].
  - 15–30 min. per multiplication gate, public key $\approx$ 1 GB.
- 2011–2013: (R)LWE-based schemes.
  - Simpler, more efficient, more versatile.
  - Optimized implementations evaluate the full AES circuit in 10–30 min. amortized.
  - Lauter et al. announce performance in the millisecond per gate range for shallow circuits.

Efficiency is improving

- 2009: breakthrough scheme by Gentry.
  - Concrete parameters unclear, probably prohibitively inefficient.
- 2010: vDGHV scheme over the integers.
  - Public key size $> 2^{60}$ bits at reasonable security levels!
- 2011: first implementations of these schemes with numerous optimizations [GH11], [CMNT11].
  - 15–30 min. per multiplication gate, public key ≈ 1 GB.
- 2011–2013: (R)LWE-based schemes.
  - Simpler, more efficient, more versatile.
  - Optimized implementations evaluate the full AES circuit in 10–30 min. amortized.
  - Lauter et al. announce performance in the millisecond per gate range for shallow circuits.

Outline

Ingredients for FHE

- Consider an integer lattice $L \subset \mathbb{Z}^n$.
  - Secret/private key: a good basis for the lattice, that can "correct large errors".
  - Public key (optional): a bad basis, or even a noisy bad basis: lets you sample a point close to the lattice, but not distinguish between a point close to the lattice and a random point.
- With this data, we can construct an encryption scheme:
  - $E_{\mathrm{pk}}(0)$ is a point close to the lattice.
  - $E_{\mathrm{pk}}(1)$ is a random point.
  - CPA secure by assumption!
  - Almost additively homomorphic.

Ingredients for FHE

- Consider an integer lattice $L \subset \mathbb{Z}^n$.
  - Secret/private key: a good basis for the lattice, that can "correct large errors".
  - Public key (optional): a bad basis, or even a noisy bad basis: lets you sample a point close to the lattice, but not distinguish between a point close to the lattice and a random point.
- With this data, we can construct an encryption scheme:
  - $E_{\mathsf{pk}}(0)$ is a point close to the lattice.
  - $E_{\mathsf{pk}}(1)$ is a random point.
  - CPA secure by assumption!
  - Almost additively homomorphic.

A more homomorphic construction

- Idea for addition: encode the message in the parity of the noise.
  - Say the pk lets you sample a point of the form $\mathbf{x} + 2\mathbf{e}$ ($\mathbf{x}$ lattice point, $\mathbf{e}$ small random error).
  - Then, encrypt $m \in \{0, 1\}$ as:

$$E_{\mathrm{pk}}(m) = \mathbf{x} + 2\mathbf{e} + (m, 0, \ldots, 0)$$

  - Still CPA, easy to decrypt.
  - And now, additively homomorphic.
- For multiplication, use an ideal lattice.

A more homomorphic construction

- Idea for addition: encode the message in the parity of the noise.
  - Say the pk lets you sample a point of the form $\mathbf{x} + 2\mathbf{e}$ ($\mathbf{x}$ lattice point, $\mathbf{e}$ small random error).
  - Then, encrypt $m \in \{0, 1\}$ as:

$$E_{\mathsf{pk}}(m) = \mathbf{x} + 2\mathbf{e} + (m, 0, \dots, 0)$$

  - Still CPA, easy to decrypt.
  - And now, additively homomorphic.
- For multiplication, use an ideal lattice.

Gentry's SHE scheme

- Public parameters: n a power of 2, $R = \mathbb{Z}[x]/(x^n + 1)$.
- Key generation returns a lattice L which is an ideal of $\mathbb{Z}[x]/(x^n + 1)$.
  - Private key is a good basis B_{sk} for L, whose fundamental parallelipiped contains the ball of radius d.
  - Public key is a bad basis B_{pk} for L (usually the HNF); with it, decisional BDD up to distance d should be hard.
- $\mathsf{Encrypt}(\mathsf{pk}, m) = 2\mathbf{e} + m \bmod B_{\mathsf{pk}}$, with $\mathbf{e}$ random such that $\|\mathbf{e}\| < \delta$.
  - Thus a ciphertext is of the form $\mathbf{x} + 2\mathbf{e} + m$ for some $\mathbf{x} \in L$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c}) = (\mathbf{c} \bmod B_{\mathsf{sk}}) \bmod 2$.
  - Correct decryption if the "noise" is of norm $< d$.
- Add and Mult are the corresponding operations in R.

Gentry's SHE scheme

- Public parameters: n a power of 2, $R = \mathbb{Z}[x]/(x^n + 1)$.
- Key generation returns a lattice L which is an ideal of $\mathbb{Z}[x]/(x^n + 1)$.
  - Private key is a good basis B_{sk} for L, whose fundamental parallelipiped contains the ball of radius d.
  - Public key is a bad basis B_{pk} for L (usually the HNF); with it, decisional BDD up to distance d should be hard.
- $\mathrm{Encrypt}(\mathrm{pk}, m) = 2\mathbf{e} + m \bmod B_{\mathrm{pk}}$, with $\mathbf{e}$ random such that $\|\mathbf{e}\| < \delta$.
  - Thus a ciphertext is of the form $\mathbf{x} + 2\mathbf{e} + m$ for some $\mathbf{x} \in L$.
- $\mathrm{Decrypt}(\mathrm{sk}, \mathbf{c}) = (\mathbf{c} \bmod B_{\mathrm{sk}}) \bmod 2$.
  - Correct decryption if the "noise" is of norm $< d$.
- Add and Mult are the corresponding operations in R.

Gentry's SHE scheme

- Public parameters: n a power of 2, $R = \mathbb{Z}[x]/(x^n + 1)$.
- Key generation returns a lattice L which is an ideal of $\mathbb{Z}[x]/(x^n + 1)$.
  - Private key is a good basis B_{sk} for L, whose fundamental parallelipiped contains the ball of radius d.
  - Public key is a bad basis B_{pk} for L (usually the HNF); with it, decisional BDD up to distance d should be hard.
- $\mathsf{Encrypt}(\mathsf{pk}, m) = 2\mathbf{e} + m \bmod B_{\mathsf{pk}}$, with $\mathbf{e}$ random such that $\|\mathbf{e}\| < \delta$.
  - Thus a ciphertext is of the form $\mathbf{x} + 2\mathbf{e} + m$ for some $\mathbf{x} \in L$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c}) = (\mathbf{c} \bmod B_{\mathsf{sk}}) \bmod 2$.
  - Correct decryption if the "noise" is of norm $< d$.
- Add and Mult are the corresponding operations in R.

Gentry's SHE scheme

- Public parameters: n a power of 2, $R = \mathbb{Z}[x]/(x^n + 1)$.
- Key generation returns a lattice L which is an ideal of $\mathbb{Z}[x]/(x^n + 1)$.
  - Private key is a good basis B_{sk} for L, whose fundamental parallelipiped contains the ball of radius d.
  - Public key is a bad basis B_{pk} for L (usually the HNF); with it, decisional BDD up to distance d should be hard.
- $\mathsf{Encrypt}(\mathsf{pk}, m) = 2\mathbf{e} + m \bmod B_{\mathsf{pk}}$, with $\mathbf{e}$ random such that $\|\mathbf{e}\| < \delta$.
  - Thus a ciphertext is of the form $\mathbf{x} + 2\mathbf{e} + m$ for some $\mathbf{x} \in L$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c}) = (\mathbf{c} \bmod B_{\mathsf{sk}}) \bmod 2$.
  - Correct decryption if the "noise" is of norm $< d$.
- Add and Mult are the corresponding operations in R.

Gentry's SHE scheme

- Public parameters: n a power of 2, $R = \mathbb{Z}[x]/(x^n + 1)$.
- Key generation returns a lattice L which is an ideal of $\mathbb{Z}[x]/(x^n + 1)$.
  - Private key is a good basis B_{sk} for L, whose fundamental parallelipiped contains the ball of radius d.
  - Public key is a bad basis B_{pk} for L (usually the HNF); with it, decisional BDD up to distance d should be hard.
- $\mathsf{Encrypt}(\mathsf{pk}, m) = 2\mathbf{e} + m \bmod B_{\mathsf{pk}}$, with $\mathbf{e}$ random such that $\|\mathbf{e}\| < \delta$.
  - Thus a ciphertext is of the form $\mathbf{x} + 2\mathbf{e} + m$ for some $\mathbf{x} \in L$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c}) = (\mathbf{c} \bmod B_{\mathsf{sk}}) \bmod 2$.
  - Correct decryption if the "noise" is of norm $< d$.
- Add and Mult are the corresponding operations in R.

Homomorphic properties of Gentry's scheme

- Addition:

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{x}_1 + 2\mathbf{e}_1 + m_1 \\ \mathbf{c}_2 &= \mathbf{x}_2 + 2\mathbf{e}_2 + m_2 \end{aligned} \Rightarrow \mathbf{c}_1 + \mathbf{c}_2 = \mathbf{x}' + 2\mathbf{e}' + m_1 + m_2$$

- Multiplication:

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{x}_1 + 2\mathbf{e}_1 + m_1 \\ \mathbf{c}_2 &= \mathbf{x}_2 + 2\mathbf{e}_2 + m_2 \end{aligned} \Rightarrow \mathbf{c}_1 \cdot \mathbf{c}_2 = \mathbf{x}' + 2\mathbf{e}' + m_1 \cdot m_2$$

with $\mathbf{e}' = 2\mathbf{e}_1 \cdot \mathbf{e}_2 + m_1\mathbf{e}_2 + m_2\mathbf{e}_1$.

  - In particular, $\|\mathbf{e}'\| \lesssim 2\sqrt{n}\|\mathbf{e}_1\| \cdot \|\mathbf{e}_2\|$.
  - The scheme supports circuits with $\approx \log_2\left(\frac{\log_2 d}{\log_2 \delta}\right)$ levels of Mult gates (somewhat homomorphic encryption).

Homomorphic properties of Gentry's scheme

- Addition:

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{x}_1 + 2\mathbf{e}_1 + m_1 \\ \mathbf{c}_2 &= \mathbf{x}_2 + 2\mathbf{e}_2 + m_2 \end{aligned} \quad \Rightarrow \mathbf{c}_1 + \mathbf{c}_2 = \mathbf{x}' + 2\mathbf{e}' + m_1 + m_2$$

- Multiplication:

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{x}_1 + 2\mathbf{e}_1 + m_1 \\ \mathbf{c}_2 &= \mathbf{x}_2 + 2\mathbf{e}_2 + m_2 \end{aligned} \quad \Rightarrow \mathbf{c}_1 \cdot \mathbf{c}_2 = \mathbf{x}' + 2\mathbf{e}' + m_1 \cdot m_2$$

  with $\mathbf{e}' = 2\mathbf{e}_1 \cdot \mathbf{e}_2 + m_1\mathbf{e}_2 + m_2\mathbf{e}_1$.
  - In particular, $\|\mathbf{e}'\| \lesssim 2\sqrt{n}\|\mathbf{e}_1\| \cdot \|\mathbf{e}_2\|$.
  - The scheme supports circuits with $\approx \log_2\left(\frac{\log_2 d}{\log_2 \delta}\right)$ levels of Mult gates (somewhat homomorphic encryption).

Outline

The DGHV Scheme (symmetric version)

- Ciphertext for $m \in \{0, 1\}$:

$$c = q \cdot p + 2r + m$$

where p is the secret key (lattice basis), q and r are randoms.

- Decryption:

$$(c \bmod p) \bmod 2 = m$$

- Parameters:

The DGHV Scheme (symmetric version)

- Ciphertext for $m \in \{0, 1\}$:

$$c = q \cdot p + 2r + m$$

where p is the secret key (lattice basis), q and r are randoms.

- Decryption:

$$(c \bmod p) \bmod 2 = m$$

- Parameters:

The DGHV Scheme (symmetric version)

- Ciphertext for $m \in \{0, 1\}$:

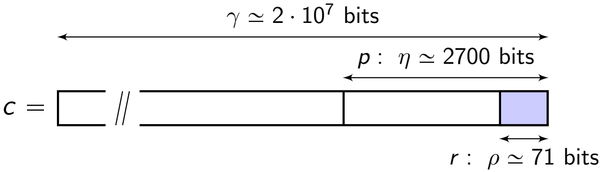
$$c = q \cdot p + 2r + m$$

where p is the secret key (lattice basis), q and r are randoms.

- Decryption:

$$(c \bmod p) \bmod 2 = m$$

- Parameters:



Homomorphic properties of vDGHV

- Addition:

$$\begin{aligned} c_1 &= q_1 \cdot p + 2r_1 + m_1 \\ c_2 &= q_2 \cdot p + 2r_2 + m_2 \end{aligned} \Rightarrow c_1 + c_2 = q' \cdot p + 2r' + m_1 + m_2$$

- Multiplication:

$$\begin{aligned} c_1 &= q_1 \cdot p + 2r_1 + m_1 \\ c_2 &= q_2 \cdot p + 2r_2 + m_2 \end{aligned} \Rightarrow c_1 \cdot c_2 = q'' \cdot p + 2r'' + m_1 \cdot m_2$$

with

$$r'' = 2r_1 r_2 + r_1 m_2 + r_2 m_1$$

  - Noise becomes twice as large.

Homomorphic properties of vDGHV

- Addition:

$$\begin{array}{l} c_1 = q_1 \cdot p + 2r_1 + m_1 \\ c_2 = q_2 \cdot p + 2r_2 + m_2 \end{array} \Rightarrow c_1 + c_2 = q' \cdot p + 2r' + m_1 + m_2$$

- Multiplication:

$$\begin{array}{l} c_1 = q_1 \cdot p + 2r_1 + m_1 \\ c_2 = q_2 \cdot p + 2r_2 + m_2 \end{array} \Rightarrow c_1 \cdot c_2 = q'' \cdot p + 2r'' + m_1 \cdot m_2$$

with

$$r'' = 2r_1 r_2 + r_1 m_2 + r_2 m_1$$

  - Noise becomes twice as large.

Public-key encryption with vDGHV

- We need to provide a "noisy" description of the ideal $p\mathbb{Z}$
- Ciphertext

$$c = q \cdot p + 2r + m$$

- Public-key: a set of τ encryptions of 0's.

$$x_i = q_i \cdot p + 2r_i$$

- Public-key encryption:

$$c = m + 2r + \sum_{i=1}^{\tau} \varepsilon_i \cdot x_i$$

for random $\varepsilon_i \in \{0, 1\}$.

Public-key encryption with vDGHV

- We need to provide a "noisy" description of the ideal $p\mathbb{Z}$
- Ciphertext

$$c = q \cdot p + 2r + m$$

- Public-key: a set of τ encryptions of 0's.

$$x_i = q_i \cdot p + 2r_i$$

- Public-key encryption:

$$c = m + 2r + \sum_{i=1}^{\tau} \varepsilon_i \cdot x_i$$

for random $\varepsilon_i \in \{0, 1\}$.

Public-key encryption with vDGHV

- We need to provide a "noisy" description of the ideal $p\mathbb{Z}$
- Ciphertext

$$c = q \cdot p + 2r + m$$

- Public-key: a set of τ encryptions of 0's.

$$x_i = q_i \cdot p + 2r_i$$

- Public-key encryption:

$$c = m + 2r + \sum_{i=1}^{\tau} \varepsilon_i \cdot x_i$$

for random $\varepsilon_i \in \{0, 1\}$.

Public-key encryption with vDGHV

- We need to provide a "noisy" description of the ideal $p\mathbb{Z}$
- Ciphertext

$$c = q \cdot p + 2r + m$$

- Public-key: a set of τ encryptions of 0's.

$$x_i = q_i \cdot p + 2r_i$$

- Public-key encryption:

$$c = m + 2r + \sum_{i=1}^{\tau} \varepsilon_i \cdot x_i$$

for random $\varepsilon_i \in \{0, 1\}$.

Security of the scheme

- As described here, reduces to the General Approximate GCD (GACD) problem: given polynomially many close multiples of p, find p.
  - Idea of the reduction: using an adversary that distinguishes $E(0)$ and $E(1)$ with significant probability, construct an algorithm that predicts the LSB of q in $q \cdot p + r$ with high probability. Conclude using binary GCD.
- In practice, we change the algorithm slightly, by adding an exact multiple of p, $x_0 = q_0 \cdot p$, in the public key.
  - Then, homomorphic addition an multiplication can be done mod x_0, keeping ciphertexts from growing exponentially.
  - The reduction is then to the Partial Approximate GCD (PACD) problem.

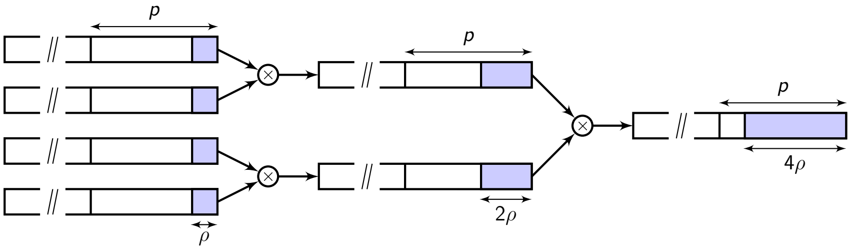
Security of the scheme

- As described here, reduces to the General Approximate GCD (GACD) problem: given polynomially many close multiples of p, find p.
  - Idea of the reduction: using an adversary that distinguishes $E(0)$ and $E(1)$ with significant probability, construct an algorithm that predicts the LSB of q in $q \cdot p + r$ with high probability. Conclude using binary GCD.
- In practice, we change the algorithm slightly, by adding an exact multiple of p, $x_0 = q_0 \cdot p$, in the public key.
  - Then, homomorphic addition an multiplication can be done mod x_0, keeping ciphertexts from growing exponentially.
  - The reduction is then to the Partial Approximate GCD (PACD) problem.

Outline

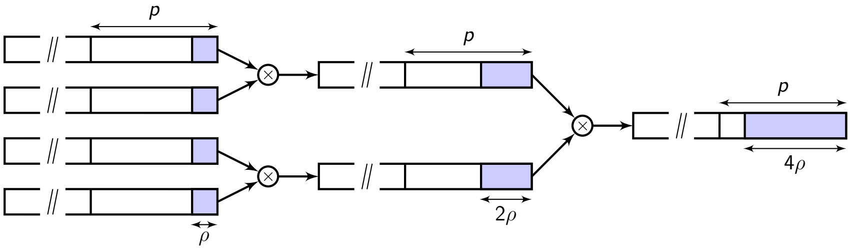
Somewhat homomorphic scheme

- The number of multiplications is limited.
  - Noise grows with the number of multiplications.
  - Noise must remain $< p$ for correct decryption.
- This is a problem with all schemes in this framework.



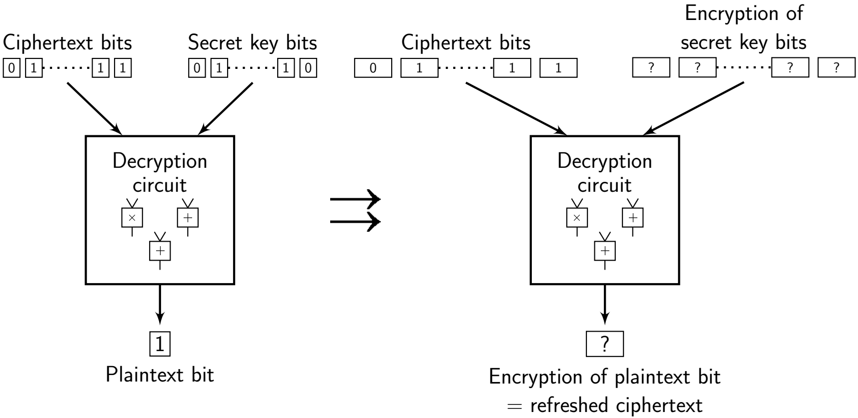
Somewhat homomorphic scheme

- The number of multiplications is limited.
  - Noise grows with the number of multiplications.
  - Noise must remain $< p$ for correct decryption.
- This is a problem with all schemes in this framework.



Solution: Bootstrapping

- Gentry's breakthrough idea: refresh the ciphertext by evaluating the decryption circuit homomorphically: bootstrapping.



Ciphertext refresh

- Refreshed ciphertext:
  - If the degree of the decryption polynomial is small enough, the resulting noise in this new ciphertext can be smaller than in the original ciphertext
- Fully homomorphic encryption:
  - Given two refreshed ciphertexts one can apply again the homomorphic operation (either addition or multiplication), which was not necessarily possible on the original ciphertexts because of the noise threshold.
  - Using this ciphertext refresh (or recryption) procedure, the number of homomorphic operations becomes unlimited and we get a fully homomorphic encryption scheme.

Ciphertext refresh

- Refreshed ciphertext:
  - If the degree of the decryption polynomial is small enough, the resulting noise in this new ciphertext can be smaller than in the original ciphertext
- Fully homomorphic encryption:
  - Given two refreshed ciphertexts one can apply again the homomorphic operation (either addition or multiplication), which was not necessarily possible on the original ciphertexts because of the noise threshold.
  - Using this ciphertext refresh (or recryption) procedure, the number of homomorphic operations becomes unlimited and we get a fully homomorphic encryption scheme.

Problems with bootstrapping

- Do we know that the encryption scheme remains secure even after publishing encryption of the secret key bits?
  - This is called circular security.
  - Only a couple of encryption schemes are proved circular secure, none of them fully homomorphic.
  - Add circular security as an ad hoc assumption.
- The noise of refreshed ciphertexts depends on the AND-depth d of the decryption circuit (it is roughly $d\rho$, where ρ is the noise of fresh ciphertexts).
  - But d can be huge! In vDGHV, it is the depth of the circuit computing $(c \bmod p) \bmod 2$ given c and the bits of p.
  - Probably impossible to set parameters making the scheme bootstrappable as is.
  - We need squashing: change the decryption algorithm to make it low depth. Quite technical.

Problems with bootstrapping

- Do we know that the encryption scheme remains secure even after publishing encryption of the secret key bits?
  - This is called circular security.
  - Only a couple of encryption schemes are proved circular secure, none of them fully homomorphic.
  - Add circular security as an ad hoc assumption.
- The noise of refreshed ciphertexts depends on the AND-depth d of the decryption circuit (it is roughly $d\rho$, where ρ is the noise of fresh ciphertexts).
  - But d can be huge! In vDGHV, it is the depth of the circuit computing $(c \bmod p) \bmod 2$ given c and the bits of p.
  - Probably impossible to set parameters making the scheme bootstrappable as is.
  - We need squashing: change the decryption algorithm to make it low depth. Quite technical.

The squashed vDGHV scheme (idea)

- Write decryption as:

$$m \leftarrow [c]_2 \oplus [\lfloor c \cdot (1/p) \rceil]_2$$

This formula can be used for ciphertext refresh if $1/p$ can be put in a compact encrypted form in the public key.

- Idea (Gentry): use secret sharing. Represent $1/p$ as a sparse subset sum:

$$\lfloor 2^\kappa / p \rceil = \sum_{i=1}^{\Theta} s_i \cdot u_i$$

with random κ-bit integers u_i, and $s_i \in \{0,1\}$. Publish the u_i's and encryptions of the s_i's.

- The decryption function can then be expressed as a polynomial of low degree (30) in the s_i's.

The squashed vDGHV scheme (idea)

- Write decryption as:

$$m \leftarrow [c]_2 \oplus [\lfloor c \cdot (1/p) \rceil]_2$$

  This formula can be used for ciphertext refresh if $1/p$ can be put in a compact encrypted form in the public key.

- Idea (Gentry): use secret sharing. Represent $1/p$ as a sparse subset sum:

$$\lfloor 2^\kappa / p \rceil = \sum_{i=1}^{\Theta} s_i \cdot u_i$$

  with random κ-bit integers u_i, and $s_i \in \{0, 1\}$. Publish the u_i's and encryptions of the s_i's.

- The decryption function can then be expressed as a polynomial of low degree (30) in the s_i's.

The squashed vDGHV scheme (idea)

- Write decryption as:

$$m \leftarrow [c]_2 \oplus [\lfloor c \cdot (1/p) \rceil]_2$$

  This formula can be used for ciphertext refresh if $1/p$ can be put in a compact encrypted form in the public key.

- Idea (Gentry): use secret sharing. Represent $1/p$ as a sparse subset sum:

$$\lfloor 2^\kappa / p \rceil = \sum_{i=1}^{\Theta} s_i \cdot u_i$$

  with random κ-bit integers u_i, and $s_i \in \{0, 1\}$. Publish the u_i's and encryptions of the s_i's.

- The decryption function can then be expressed as a polynomial of low degree (30) in the s_i's.

A little game

- In 2012, the best paper award of a minor Indian conference went to an "improvement" of the vDGHV scheme that goes basically like this:
  - Only two public key elements $x_0 = q_0 \cdot p$, $x_1 = q_1 \cdot p + 2r_1$.
  - Encrypt m as $c = m + 2r'_0 + r'_1 x_1 \bmod x_0$ for small random r'_0, r'_1.
  - Decrypt as before.
- Game for tomorrow:
  - Show that one can decrypt any ciphertext with the public key alone!
  - Hint: this involves lattice reduction in very small dimension.

A little game

- In 2012, the best paper award of a minor Indian conference went to an "improvement" of the vDGHV scheme that goes basically like this:
  - Only two public key elements $x_0 = q_0 \cdot p$, $x_1 = q_1 \cdot p + 2r_1$.
  - Encrypt m as $c = m + 2r'_0 + r'_1 x_1 \bmod x_0$ for small random r'_0, r'_1.
  - Decrypt as before.
- Game for tomorrow:
  - Show that one can decrypt any ciphertext with the public key alone!
  - Hint: this involves lattice reduction in very small dimension.