New Trend in Cryptography – Master SCCI

Vanessa Vitse

2015-2016

Contents

1 Homomorphic encryption

Most public key cryptosystems rely on algebraic structures: groups, rings, lattices, polynomial systems... As a consequence they often have a *homomorphic property*, as in the RSA example.

Example. *Let (N, e) be the public key of a RSA cryptosystem, which operates on the ring $\mathbb{Z}/N\mathbb{Z}$. If $m_1, m_2 \in \mathbb{Z}/N\mathbb{Z}$ are two plaintexts, the corresponding ciphertexts are $c_1 = m_1^e \mod N$ and $c_2 = m_2^e \mod N$ respectively. So $c_1.c_2 = m_1^e.m_2^e = (m_1.m_2)^e \mod N$, i.e. $c_1.c_2$ is the ciphertext corresponding to the plaintext $m_1.m_2$: the encryption map is a group morphism from $((\mathbb{Z}/N\mathbb{Z})^*, \times)$ to itself.*

This is a special case of a property called malleability: an encryption algorithm is *malleable* if, given an encryption c of a plaintext m, it is possible for an adversary to generate another ciphertext c' which decrypts to $f(m)$ for a known function f, without necessarily knowing or learning m. This is usually considered as an undesirable property, and malleable cryptosystems usually require extra precautions such as message padding to avoid e.g. forgery attacks.

Exercise 1. Find the homomorphic property of the ElGamal encryption scheme.

But the homomorphic property could actually be an advantage. A growing trend is to delegate data storage to remote servers (the "cloud"), and the same is true for computing power. For privacy reasons, it is sensible to store only encrypted data on remote servers, but then the processing of data is problematic:

- either the user has to retrieve the data on his own servers, decrypt it, process it, re-encrypt the result and upload it to the remote servers, but this prevents the use of remote computing power;
- or the user has to entrust a remote server with his private key, but this threatens his privacy and security.

Fully homomorphic encryption aims at solving this dilemma, and was in fact proposed by Rivest, Adleman and Dertouzos back in 1978; but the first secure (though impractical) implementation was found by C. Gentry in 2009. For the following definition, we consider a subset $\mathcal{F}$ of the set of all functions with fixed number of variables, i.e. a subset of the union for $n \in \mathbb{N}$ of the spaces of functions $\{0,1\}^n \to \{0,1\}$.

Definition 1.1. *A homomorphic encryption scheme with respect to $\mathcal{F}$ consists of the following algorithms:*

- `KeyGen`*, which takes as input a security parameter λ and outputs a secret key (in a symmetric setting) or a pair of public and private keys (in an asymmetric setting);*
- `Encrypt`*, which takes as input a (secret or public) key and a plaintext and outputs the corresponding ciphertext;*
- `Decrypt`*, which takes as input a (secret or private) key and a ciphertext and outputs the corresponding plaintext;*
- `Evaluate`*, which takes as input a function of $\mathcal{F}$ and a number of ciphertexts (and possibly a public key) and outputs a ciphertext, such that*

$$\texttt{Decrypt}(sk, \texttt{Evaluate}(f, c_1, \ldots, c_n, pk)) = f(\texttt{Decrypt}(sk, c_1), \ldots, \texttt{Decrypt}(sk, c_n))$$

  where sk and pk are a pair of private/public keys.

Such a scheme has the compact ciphertexts *property if the outputs of* `Evaluate` *have the same length as the outputs of* `Encrypt` *and are as hard to decrypt. It is called* efficient *if the four algorithms have polynomial complexity in λ; for* `Evaluate`*, its complexity must also be polynomial in terms of the "size" of the function f (see below).*

An encryption scheme is fully homomorphic (Fully Homomorphic Encryption scheme, FHE) if it has compact ciphertexts, is efficient and $\mathcal{F}$ consists of all functions.

In other words, (fully) homomorphic encryption allows a non-trusted party to operate on encrypted data without loss of confidentiality. Note that in the context of remote storage and computing, the distinction between the symmetric and the asymmetric settings is blurred since it is often the same entity that encrypts and decrypts its own data.

Some remarks:

- By security parameter λ, we mean that the attacks on the encryption scheme must have a complexity that is exponential in λ.
- The purpose of the compact ciphertexts condition is to ensure that it is indeed the remote server that does the computation. Otherwise, it is possible to construct "trivial" FHE schemes, where the output of $\texttt{Evaluate}(f, c_1, \ldots, c_n, pk)$ is some encryption of f and its arguments that basically tells the function `Decrypt` to decrypt $c_1, \ldots, c_n$ and compute f on these decrypted values.
- The size of the function f must measure not only its running time but also the length of its description. A way of doing that is to consider the size of a boolean circuit that computes f, i.e. its number of elementary AND, OR and NOT gates. We will see that it is actually a quite accurate measure in our context.
- In the above description, it is also possible to encrypt in some way the description of the function $f \in \mathcal{F}$, so that the remote server not only does not know on what data it operates, but does not even know what computation it is doing.

FHE can be used for many applications; we just list a few of them.

- Processing of sensitive data: in this typical scenario, the user asks a company to perform some analysis (that the user himself is not able or willing to perform) on his data. Homomorphic encryption allows to preserve the privacy of the processed data. For instance, one can imagine cloud services that provide DNA analysis for medical institutions and law enforcement authorities while maintaining the confidentiality of DNA samples.
- Anonymous data processing: multiple users can send sensitive data, encrypted with some public key, to a remote server. Using homomorphic evaluation, it is possible to extract statistical information on the data set in encrypted form, without the remote server learning anything about any user's data. The encrypted anonymized results (and only those) are then delivered to the final recipient, who can decrypt them with the private decryption key.
- e-voting: each voter encrypts his/her ballot with a public encryption key provided by the organizers. Because of this encryption, there is no need to secure the anonymity of the voters; the encrypted ballots can be made public, and in particular every voter can check that his/her vote is taken into account. Also, every user can check the encrypted result of the vote using homomorphic evaluation. The organizers can now decrypt this result. Of course, other precautions must be taken to ensure that the organizers can only decrypt the final result and not the individual votes.

However, FHE is not well-suited for databases applications. For instance, encrypted search on a large database (like those maintained by search engines) is problematic: it is not possible to ensure that the server learns nothing about the result of a query without the whole database being scanned.

In the following, we will only consider schemes where the data is encrypted *bitwise*, i.e. each bit is encrypted separately, independently from the others. In other words, we deal with block ciphers in ECB mode with a block size of one bit. Then in order to evaluate any function, we just have to be able to evaluate the addition and the multiplication modulo 2, since in $\mathbb{Z}/2\mathbb{Z} \simeq \{0,1\}$ we have $\mathrm{AND}(a,b) = a.b$, $\mathrm{OR}(a,b) = a + b + a.b$ and $\mathrm{NOT}(a) = 1 + a$: an encryption scheme is fully homomorphic as soon as we can add and multiply the underlying plaintexts.

Of course, a secure encryption function $\{0,1\} \to \{0,1\}^*$ cannot be deterministic: a very short brute force attack would decrypt any data immediately! The encryption function must be probabilistic, and ideally, for security no identical bit should have the same ciphertext. In practice, we ask that the probability that two identical bits have the same encryption is negligible unless the quantity of data is very important (depending on the expected security level). This implies that each bit is translated as a rather long string.

Exercise 2 . A given FHE scheme encrypts probabilistically each bit as an n-bit ciphertext; for simplicity, we assume that the ciphertexts have a uniform probability distribution inside the space of n-bit strings. We want to store one Terabit of (plaintext) data. Approximately, what is the value of n after which the probability of having two bits identically encrypted becomes negligible? What is the size of the encrypted data?

Actually, current FHE schemes have much larger expansion factors, between 1 000 and 10 000. Obviously they are rather impractical and cannot be deployed for real-world applications.

Note that instead of encoding bitwise, we could consider a different alphabet from $\{0,1\} \simeq \mathbb{Z}/2\mathbb{Z}$. For instance, we could work with $\{0,\ldots,p-1\} \simeq \mathbb{Z}/p\mathbb{Z}$, and in order to be fully homomorphic it is sufficient to be able to add and multiply ciphertexts. But probabilistic encryption is still necessary, even with large p. Indeed, the existence of an Evaluate algorithm means that FHE schemes are highly malleable by design, and this can be exploited by an attacker: Boneh and Lipton showed in 1996 that any deterministic, FHE scheme can be broken in sub-exponential time.

Exercise 3 . Show that any function $f : (\mathbb{Z}/p\mathbb{Z})^n \to \mathbb{Z}/p\mathbb{Z}$ (with p a prime) admits a polynomial expression (hint: use Lagrange interpolation).

2 Somewhat homomorphic encryption

Secure FHE schemes are quite difficult to devise (the first, impractical construction appeared thirty years after the definition was proposed). On the other hand, many public-key cryptosystems are already homomorphic with respect to only one operation (addition or multiplication). A first step toward FHE is the construction of cryptosystems that allow a restricted number of manipulations on the ciphertexts.

2.1 Boneh-Goh-Nissim encryption scheme

In 2005, Boneh, Goh and Nissim proposed a cryptosystem that allows any evaluation of homogeneous polynomials of degree less than 2. It is based on elliptic curves whose group order are hard to factorize and relies on pairings for the multiplication step.

1. Key generation: select two large primes q_1 and q_2 and let $n = q_1 q_2$. Find the smallest integer $\ell \in 3\mathbb{N}$ such that $p = \ell n - 1$ is prime (and equals 2 mod 3). Then the elliptic curve $E : y^2 = x^3 + 1$, defined over $\mathbb{F}_p$, is super-singular and has exactly ℓn rational points. Let P be a random point of order n and $Q = [kq_2]P$ a random multiple of P of order q_1. Since E is super-singular, there

exists a pairing (the modified Weil pairing) $e : \langle P\rangle \times \langle P\rangle \to \mathbb{F}_{p^2}$. The public key is (n, E, P, Q, e) and the private key is q_1. The message space consists of integers in the range $\{-T/2, \dots, T/2\}$, with $T < q_2$, and the cryptogram space is $E(\mathbb{F}_p) \cup \mathbb{F}_{p^2}$; there are thus two distinct types of ciphertexts.

2. Encryption: to encrypt a message m, choose a random $r \in \{1, n-1\}$ and output the ciphertext $C = [m]P + [r]Q \in E(\mathbb{F}_p)$.

3. Addition: if $C_1 = [m_1]P + [r_1]Q$ and $C_2 = [m_2]P + [r_2]Q$ are ciphertexts in $E(\mathbb{F}_p)$ that encrypt m_1 and m_2 respectively, then $C_1 + C_2 + [r]Q \in E(\mathbb{F}_p)$ (for a random $r \in \{1, n-1\}$) is a re-randomized encryption of $m_1 + m_2$.

4. Multiplication: let C_1 and C_2 be two ciphertexts in $E(\mathbb{F}_p)$ that encrypt m_1 and m_2 respectively. Then the encryption of the product m_1m_2 is $C = e(C_1, C_2).e(Q, Q)^r \in \mathbb{F}_{p^2}$ (for a random $r \in \{1, n-1\}$).

5. Addition (again): if C_1 and C_2 are ciphertexts in $\mathbb{F}_{p^2}$ that encrypts m_1 and m_2 respectively, then $C = C_1.C_2.e(Q, Q)^r \in F_{p^2}$ (for a random $r \in \{1, n-1\}$) is a re-randomized encryption of $m_1 + m_2$

6. Decryption: if the ciphertext C is in $E(\mathbb{F}_p)$, compute $\tilde{P} = [q_1]P$, $\tilde{C} = [q_1]C$, and output $m' = \log_{\tilde{P}}(\tilde{C})$. If C is in $\mathbb{F}_{p^2}$, compute $\tilde{g} = e(P, P)^{q_1}$, $\tilde{C} = C^{q_1}$, and output $m' = \log_{\tilde{g}}(\tilde{C})$.

We leave it as an exercise to show the correction of this cryptosystem. Only one multiplication is supported because it changes the nature of the ciphertext, and there is no pairing on finite fields to pursue this construction further. The main idea is that the ciphertexts live in an order n group (generated by either P or $e(P, P)$) but are only well-defined modulo the subgroup of order q_1 (generated by either Q or $e(P, Q)$); the obfuscation of the plaintexts is achieved by choosing random representatives. The security of this scheme is based on the difficulty of the following subgroup decision problem: *given a cyclic group G of known order n and a proper subgroup $H \subset G$, determine if an element $g \in G$ is in the subgroup H.* For cyclic groups, this is easy if the factorisation of n if known, but it is not clear if it can be solved without factoring the group order. To see why this is actually the problem the security relies on, observe that the discrete log of $\tilde{C}$ in base $\tilde{P}$ can be recovered as the smallest positive integer m' such that $C - [m']P$ belongs to the subgroup generated by Q (and this adapts trivially to a ciphertext in $\mathbb{F}_{p^2}$).

The drawback of Boneh-Goh-Nissim construction is the decryption step, which requires the computation of a discrete logarithm in the subgroup of order q_2 generated by $\tilde{P}$ or $\tilde{g}$. The size of p must be such that the discrete log problem in $E(\mathbb{F}_p)$ and in $\mathbb{F}_{p^2}$ is hard; otherwise, n can be factored by computing $\gcd(n, \log_P(Q))$ or $\gcd(n, \log_{e(P,P)}(e(P, Q)))$. But then the DLP in the subgroup of order q_2 is also hard, even if its size is smaller by a factor of about 2. Indeed, the index calculus method available on $\mathbb{F}^*_{p^2}$ is not easier in a subgroup than in the whole multiplicative group, and if q_2 is small enough so that generic algorithms are efficient then n cannot be hard to factor using sieve algorithms. The solution is that only small values of the plaintext are allowed and can be deciphered: the messages lie not in $\mathbb{Z}/q_2\mathbb{Z}$, but in an interval $[-T/2, T/2]$ whose size T is small enough so that generic algorithms (e.g. Pollard rho) can solve the DLP, with complexity in $O(\sqrt{T})$.

But this means that this scheme is not completely homomorphic. Indeed, one can add and perform one multiplication on the ciphertexts, but decryption will be possible only if the resulting plaintext remains relatively small: the larger the plaintext, the harder the decryption. This is particularly an issue for the multiplication. One workaround is to start with even smaller messages, in order that decryption remains possible unless many operations are performed. This works well if the number of requested additions is known in advance.

2.2 van Dijk-Gentry-Halevi-Vaikuntanathan somewhat homomorphic encryption scheme over the integers

Boneh, Goh and Nissim's construction lead to the recognition of the importance of *somewhat homomorphic encryption schemes* (SHE), which are basically the same thing as FHE except that only a limited number of operations (additions and multiplications) can be performed before either decryption fails or returns a wrong answer. In other word, a SHE scheme can only evaluate functions whose complexity, expressed as a number of logical gates, does not exceed a certain value D. We will start by studying the SHE scheme proposed by van Dijk, Gentry, Halevi and Vaikuntanathan in 2009, which is conceptually simple (it only uses elementary arithmetic) yet secure, but blatantly impractical. The encryption is bitwise: the message space is thus $\{0,1\} \simeq \mathbb{Z}/2\mathbb{Z}$ and the ciphertexts are integers. The symmetric version works as follows; its security depends on a parameter λ.

1. Key generation: the secret key is a random λ^2-bit odd integer p.
2. Encryption: to encrypt a message $m \in \{0,1\}$, choose a random λ^5-bit integer q, a random λ-bit integer r, and output $c = m + 2r + pq$.
3. Decryption: to decrypt a ciphertext c, compute $m = (c \bmod p) \bmod 2$, where the notation $c \bmod p$ stands for the only integer in $(-p/2, p/2]$ congruent to c modulo p.
4. Evaluation: addition and multiplication of underlying plaintexts are simply given by the addition and the multiplication of the ciphertexts.

In this scheme, the ciphertexts are "near-multiple" of p, and it is the parity of the "noise" $m+2r$ that allows to recover the plaintext. As regards the Evaluate procedure, we check that if $c = m + 2r + pq$ and $c' = m' + 2r' + pq'$, then

$$c + c' = (m + m') + 2(r + r') + p(q + q'), \text{ and}$$

$$cc' = mm' + 2(rm' + r'm + 2rr') + p(qm' + 2qr' + q'm + 2q'r + qq').$$

The difference with a multiple of p has the correct parity, but the noise has increased, especially in the case of a multiplication. However, for the decryption to work this noise has to be smaller than $p/2$ in absolute value; to put it another way, in the above formula we need the multiple of p on the right to be the *closest* to $c + c'$, resp. cc', to be sure that the deciphered value is the correct one. This is why this scheme is only somewhat homomorphic: the more operations we perform, the noisier the ciphertexts become, until decryption yields spurious results.

We can give a safety bound on the number of allowed operations. Indeed, it is not difficult to see that addition increases the size of the noise by at most one bit, whereas multiplication roughly adds the size of the noises. Since "fresh" ciphertexts have a λ-bit noise, we can multiply approximatively λ of them before the noise reaches a size of λ^2 bits, comparable with p.

The security of this cryptosystem is based on the *approximate GCD problem*: recovering p from many "near multiples" of p. With the above parameters, the complexity of the best known attacks is exponential in the security parameter λ. Note however that this problem has garnered attention only recently, so it is conceivable that much better algorithms for this problem will appear in the future. The obvious drawback of van Dijk-Gentry-Halevi-Vaikuntanathan construction is the fact that the ciphertexts are much larger than the plaintexts; furthermore, the size of the ciphertexts increases with the number of performed operations (linearly in the number of multiplications).

Exercise 4 . Find the three-digit approximate GCD of the numbers 195051, 257797, 328385 and 360776 (hint: test all possible small noises on a pair of numbers until their gcd is large enough, then check with the remaining integers).

This scheme can be modified to become an asymmetric one. The ciphertexts are still near-multiple of a secret integer p, with the message being hidden in the parity of the noise; in particular, evaluation and decryption are the same as in the above setting.

1. Key generation: the private key is a random λ^2-bit odd integer p. The public key consists of a λ^5-size list $\{x_1, \dots, x_N\}$ of "encryptions of zero", i.e. random integers of the form $2r + pq$ with r and q of size λ bits and λ^5 bits respectively.

2. Encryption: to encrypt a message $m \in \{0, 1\}$, choose a random subset $S \subset \{1, \dots, N\}$ and a λ-bit integer r, and output $c = m + 2r + \sum_{i \in S} x_i$.

Since the x_i have a small, even noise, the "fresh" ciphertext c still has a small noise, with the correct parity. Now the security of this public-key version is a bit more difficult to analyze but still relies on the approximate GCD problem.

It is possible to modify slightly this scheme in order to avoid the important increase in the size of the ciphertexts. We add to the public key an element x_0 which is an exact multiple of p (instead of another near-multiple). Then we can reduce each ciphertext modulo x_0. This reduction amounts to subtracting a multiple of x_0 to the ciphertext, but this does not modify the noise since x_0 is an exact multiple of p, so the result is correct. Of course the price to pay is that this *partial* approximate GCD problem becomes easier, but it can still be secure if the parameters are chosen accordingly (and obviously x_0 should be hard to factorize).

2.3 Learning with errors and the Gentry-Sahai-Waters encryption scheme

The *Learning With Errors* problem (LWE), first introduced by Regev in 2005, is a versatile tool for constructing cryptosytems. In particular it is the basis for the most promising (but still inefficient. . .) SHE schemes today. We will not give in this lecture a precise analysis of its security, but it has been shown by Brakerski, Langlois, Peikert, Regev and Stehlé that LWE is at least as secure as worst-case instances of standard lattice problems.

2.3.1 Error distributions

In somewhat homomorphic encryption schemes, we often need to sample according to "error" probability distributions on $\mathbb{Z}$ (e.g. the "noise" in DGHV integer scheme) or $\mathbb{Z}/p\mathbb{Z}$, namely, we will need random variables which stay "close" to zero in some adequate sense.

On $\mathbb{Z}$, a common choice is the discrete Gaussian distribution. Recall that the continuous Gaussian distribution (or normal distribution) is defined by its probability density function

$$f_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}},$$

where the standard deviation σ controls the "width" of the distribution. In the corresponding discrete distribution, the probability of obtaining an integer n is proportional to $f_\sigma(n)$. More precisely, a integer-valued random variable X follows the discrete Gaussian probability if for all $n \in \mathbb{Z}$, $P(X = n) = f_\sigma(n)/c$ where $c = \sum_{k\in\mathbb{Z}} f_\sigma(k)$. In order to obtain a $\mathbb{Z}/p\mathbb{Z}$-valued random variable, we just have to consider the values of X modulo p. Note that this construction can actually be carried out starting with any continuous distribution on $\mathbb{R}$.

If the standard deviation of X is small compared to p, then with high probability, $X(\omega)$ is comprised between $-p/2$ and $p/2$, so no modular reductions are actually needed. On the other hand, when σ grows to infinity, the discrete Gaussian distribution modulo p converges to the uniform distribution on $\mathbb{Z}/p\mathbb{Z}$.

2.3.2 LWE

We define on $(\mathbb{Z}/p\mathbb{Z})^N$ the "usual" bilinear form

$$\begin{array}{rcl} (\mathbb{Z}/p\mathbb{Z})^N \times (\mathbb{Z}/p\mathbb{Z})^N & \to & \mathbb{Z}/p\mathbb{Z} \\ (a,b) & \mapsto & \langle a,b\rangle = \sum_{i=1}^{N} a_i b_i \end{array}$$

Let χ_α be an error probability distribution on $\mathbb{Z}/p\mathbb{Z}$, i.e. the reduction modulo p of a centered distribution on $\mathbb{Z}$, with standard deviation α. Let $s \in (\mathbb{Z}/p\mathbb{Z})^N$ be a secret vector. We consider many couples of the form $(a_k, b_k) \in (\mathbb{Z}/p\mathbb{Z})^N \times \mathbb{Z}/p\mathbb{Z}$ where a_k is chosen uniform randomly in $(\mathbb{Z}/p\mathbb{Z})^N$ and $b_k = \langle a_k, s\rangle + e_k$, with $e_k \in \mathbb{Z}/p\mathbb{Z}$ being sampled with respect to the distribution χ_α.

The computational LWE problem is:

Given many couples $(a_k, b_k) = (a_k, \langle a_k, s\rangle + e_k)$, recover s.

The decisional LWE problem consists of distinguishing the distribution (a_k, b_k) as above from the uniform distribution on $(\mathbb{Z}/p\mathbb{Z})^N \times \mathbb{Z}/p\mathbb{Z}$.

The parameter α clearly plays an important role in the difficulty of these problem. If $\alpha = 0$, there are no errors, and recovering s from (more than) n couples just amounts to solving a linear system of equations. If α is very small, then e_k is often zero and the resolution is still easy. On the other hand, if α is of the magnitude of p or bigger, then χ_α is very close to the uniform probability distribution and solving the decisional problem becomes impossible. In the following α will be in-between, sufficiently big so that the LWE problem is computationnally hard, but otherwise as small as possible.

It turns out that there exist reductions between the computational and decisional versions. Indeed, assume first that we have access to a solver for the computational problem, and that we are given samples $(a_k, b_k) \in (\mathbb{Z}/p\mathbb{Z})^{n+1}$ which are either uniformly or LWE-distributed. The computational-LWE solver outputs a corresponding vector s. Then the sequence $(b_k - \langle a_k, s\rangle)$ is distributed either according to χ_α or uniformly on $\mathbb{Z}/p\mathbb{Z}$, and if α is not to big it is not difficult to distinguish between the two cases.

On the other hand, assume that we have access to a solver for the decisional problem and that we are given many couples of the form $(a_k, b_k) = (a_k, \langle a_k, s\rangle + e_k)$. Then we can try to guess each cooordinate of $s = (s_1, \dots, s_n)$ as follows. For all $i \in \{1, \dots, n\}$, let $u_i = (0, \dots, 0, 1, 0, \dots, 0)$ be the i-th vector of the canonical basis. For all $t \in \mathbb{Z}/p\mathbb{Z}$, we consider the sequence $(a'_k, b'_k) = (a_k + c_k u_i, b_k + tc_k)$ where the c_k are uniformly distributed. Then

$$\langle a'_k, s\rangle = \langle a_k, s\rangle + c_k s_i = b_k - e_k + c_k s_i = b'_k - e_k + c_k(s_i - t).$$

If $t = s_i$, then $b'_k = \langle a'_k, s\rangle + e_k$ and the sequence is LWE-distributed; otherwise it is uniformly distributed. We can thus recover s with pn calls to the decisional solver (for comparison, a brute-force attack on s requires $\approx p^n$ trials).

Another feature of the LWE problem is random self-reducibility, which states that it is as difficult in the worst case as in the average case. Indeed, we can transform any specific LWE instance $(a_k, b_k) = (a_k, \langle a_k, s\rangle + e_k)$ into another, random instance by choosing a uniform random $t \in (\mathbb{Z}/p\mathbb{Z})^n$ and computing $(a_k, \langle a_k, t\rangle + b_k) = (a_k, \langle a_k, s + t\rangle + e_k)$. Using this transformation, any instance of LWE reduces to a random instance, implying that the wort-case complexity is the same as the average-case complexity.

The LWE problem can also be expressed using matrices. The equalities $b_k = \langle a_k, s\rangle + e_k$, $k \in \{1, \dots, m\}$ can be summed up as $b = As + e$, where

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_k \\ \vdots \\ b_m \end{pmatrix}, \quad A = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{k,1} & \dots & a_{k,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{pmatrix}, \quad e = \begin{pmatrix} e_1 \\ \vdots \\ e_k \\ \vdots \\ e_m \end{pmatrix}.$$

Alternatively, if we define $s' = (s, 1) \in (\mathbb{Z}/p\mathbb{Z})^{n+1}$ and $A' = (A| - b) \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, we obtain $A's' = e$; since e is small, this can restated by saying that s' is **approximately in the kernel** of A'. The last coordinate of s' is 1, but this can be relaxed by setting $s'' = \lambda s'$ and $A'' = (\lambda)^{-1}A'$ for $\lambda \in (\mathbb{Z}/p\mathbb{Z})^\times$.

In this context, the computational LWE problem becomes:

Given $A \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, find an approximate kernel vector of A, i.e. s such that As is small

and the decisional problem consists of distinguishing matrices admitting such vectors from uniformly random one.

2.3.3 Regev encryption scheme

It is not difficult to turn LWE in a symmetric encryption scheme. The secret key is a vector $s \in (\mathbb{Z}/p\mathbb{Z})^N$ where the values of p and N depend on the desired security level. To encrypt a bit $m \in \{0, 1\}$, one selects a uniformly random vector $a \in (\mathbb{Z}/p\mathbb{Z})^N$, a random small element $r \in \mathbb{Z}/p\mathbb{Z}$ distributed according to χ_α (the “noise”), and outputs $(a, \langle a, s\rangle + r + m\lfloor p/2\rfloor\rangle)$. Decryption is quite simple: on receiving (a, b), one computes $b - \langle a, s\rangle$ and outputs 0 or 1 depending on whether the result is closer to 0 or $p/2$. This simple scheme is naturally (somewhat) homomorphically additive: if (a_1, b_1) and (a_2, b_2) encrypt the bits m_1 and m_2, it is easy to see that $(a_1 + a_2, b_1 + b_2)$ encrypts $m_1 + m_2$, but with a potentially larger noise. As before, decryption fails when the noise becomes too large, more precisely when the sum of the noises exceed $p/4$ in absolute value.

This partial homomorphic property allows to construct an asymmetric version; we describe it using the matrix representation.

1. Key generation: the private key is a random $s \in (\mathbb{Z}/p\mathbb{Z})^{n+1}$ whose last coordinate is equal to 1. The public key consists of a matrix $A \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, constructed as above, such that $As = e \in (\mathbb{Z}/p\mathbb{Z})^m$ has small coefficients (distributed according to χ_α).

2. Encryption: to encrypt a message $m \in \{0,1\}$, choose a uniformly random vector $r \in \{0,1\}^m$ and output $c = {}^tAr + m\lfloor p/2 \rfloor u_{n+1}$, where u_{n+1} is the last vector of the canonical basis of $(\mathbb{Z}/p\mathbb{Z})^{n+1}$.

3. Decryption: to decrypt c, compute $\langle c, s\rangle = \langle {}^tAr + m\lfloor p/2 \rfloor u_{n+1}, s\rangle = \langle r, As\rangle + m\lfloor p/2 \rfloor \langle u_{n+1}, s\rangle = \langle r, e\rangle + m\lfloor p/2 \rfloor$. Since e is small and the coefficients of r are only zeroes and ones, the scalar product $\langle r, e\rangle$ is also small, so $m = 0$ or 1 according to whether $\langle c, s\rangle$ is closer to 0 or $p/2$.

Actually, each row of A can be seen as an encryption of zero, and left multiplication by ${}^tr$ corresponds to taking the sum of a subset of them. This is completely similar to what we have done for the DGHV integer scheme.

The difficulty is to make this scheme (somewhat) homomorphically multiplicative. This has been first achieved by Brakerski and Vaikunthanatan using a relinearization technique, at the cost of longer keys; we will not give the details here but present a conceptually simpler scheme instead.

2.3.4 Approximate eigenvectors and the GSW encryption scheme

Approximate eigenvectors

The main idea behind the Gentry-Sahai-Waters scheme (2013) is to use **approximate eigenvectors**. Informally, it works as follows. Let v be a secret vector in $(\mathbb{Z}/p\mathbb{Z})^m$. Assume that for each small integer λ, we are able to construct a random-looking matrix $C \in \mathcal{M}_m(\mathbb{Z}/p\mathbb{Z})$ such that v is an approximate eigenvector of C, i.e. $Cv = \lambda v + e$ where the error vector e has small coefficients (distributed according to some error probability distribution). The matrix C is the ciphertext corresponding to the plaintext λ, and recovering λ knowing C and v is easy. Is such a system somewhat homomorphic? If C, C' are encryptions of λ and λ', then

$$(C + C')v = Cv + C'v = \lambda v + e + \lambda' v + e' = (\lambda + \lambda')v + (e + e'),$$

which decrypts as $\lambda + \lambda'$ as long as the errors are small enough. On the other hand,

$$(CC')v = C(\lambda' v + e') = \lambda' Cv + Ce' = \lambda'(\lambda v + e) + Ce' = (\lambda'\lambda)v + (\lambda' e + Ce').$$

This decrypts as $\lambda\lambda'$ only if $\lambda' e + Ce'$ is small, which implies that λ' must be small, and more importantly, C must have small coefficients.

Thus in order to obtain a SHE scheme from this idea, we need to be able to construct matrices with small coefficients and prescribed approximate eigenvectors. But even then, if we do matrix multiplications the coefficients will soon become too large. If we want to perform a non negligible number of homomorphic multiplications, we need a way to "flatten" ciphertexts, i.e. a method to transform a matrix C with approximate eigenvector v into another matrix C', with small coefficients, and same approximate eigenvector and eigenvalue.

Bit expansion and flattening

Let n be a positive integer, $k = \lfloor \log p \rfloor$ and $m = (n+1)(k+1)$. Let $G \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$ be the "gadget matrix"

$$G = \begin{pmatrix} 2^k & 0 & \dots & 0 \\ 2^{k-1} & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 2^0 & 0 & \dots & 0 \\ 0 & 2^k & \dots & 0 \\ 0 & 2^{k-1}1 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 2^0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 2^k \\ 0 & 0 & & 2^{k-1} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 2^0 \end{pmatrix}$$

The map $\mathcal{M}_m(\mathbb{Z}/p\mathbb{Z}) \to \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, $M \mapsto MG$ is clearly onto. Indeed, for any $N \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, there exists a matrix $M \in \mathcal{M}_{m,m}(\mathbb{Z}/p\mathbb{Z})$ whose coefficients are only 0 and 1 and such that $MG = N$; this matrix is obtained by replacing each coefficient of N by its binary expansion. We denote by G^{-1} this bit-expansion transformation, so that for all $N \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$ we have $(G^{-1}(N))G = N$. We give an example for $p = 7$ and $n + 1 = 2$:

$$G = \begin{pmatrix} 4 & 0 \\ 2 & 0 \\ 1 & 0 \\ 0 & 4 \\ 0 & 2 \\ 0 & 1 \end{pmatrix} \qquad N = \begin{pmatrix} 2 & 4 \\ 5 & 3 \\ 0 & 5 \\ 1 & 6 \\ 3 & 0 \\ 1 & 4 \end{pmatrix} \qquad G^{-1}(N) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Now let $C \in \mathcal{M}_m(\mathbb{Z}/p\mathbb{Z})$ be a matrix having an approximate eigenvector x, i.e. $e = Cx - \lambda x$ has small coefficients. Assume furthermore that x is in the image of G, i.e. there exists $s \in (\mathbb{Z}/p\mathbb{Z})^{n+1}$ such that $x = Gs$. Then

$$G^{-1}(CG)x = (G^{-1}(CG))Gs = CGs = Cx = \lambda x + e,$$

which means that x is also an approximate eigenvector of $C' = G^{-1}(CG)$, with the same eigenvalue and error term. Furthermore the coefficients of C' are small by design, since they are only 0's and 1's. Thus the "flattening" map $C \mapsto G^{-1}(CG)$ gives us the possibility to work only with matrices having small coefficients, while preserving the approximate eigenvectors.

GSW scheme

We can now describe the approximate eigenvector scheme of Gentry, Sahai and Waters. We first choose parameters $p, n, m = (n+1)\lceil \log p \rceil$ and α such that the LWE problem in $(\mathbb{Z}/p\mathbb{Z})^n$ with error distribution χ_α is difficult. The ciphertext is $\mathcal{M}_m(\{0,1\}) \subset \mathcal{M}_m(\mathbb{Z}/p\mathbb{Z})$. We also fix a message space $\mathcal{M} \subset \mathbb{Z}/p\mathbb{Z}$ (usually $\{0,1\}$, see discussion below).

1. Key generation: the private key is a random vector $s \in (\mathbb{Z}/p\mathbb{Z})^{n+1}$ whose last coordinate is equal to 1. The public key consists of a matrix $A \in \mathcal{M}_{m,n+1}(\mathbb{Z}/p\mathbb{Z})$, constructed as above, such that $As = e \in (\mathbb{Z}/p\mathbb{Z})^m$ has small coefficients (distributed according to χ_α).

2. Encryption: to encrypt a message $\lambda \in \mathcal{M}$, choose a uniformly random matrix $R \in \mathcal{M}_m(\{0,1\})$ and output $C = G^{-1}(RA + \lambda G)$.

The key generation is exactly the same as in Regev encryption scheme. The encryption is also similar: in fact, each row of RA is a different encryption of 0 according to Regev scheme, which is then used to hide the message λ. The important point is that $v = Gs$ is an approximate eigenvector, with eigenvalue λ, of the "fresh" ciphertexts thus created. Indeed,

$$Cv = CGs = (RA + \lambda G)s = \lambda v + Re,$$

and since the coefficients of R are only 0's and 1's, the coefficients of Re are still small.

Exercise 5. Show that λ is the only approximate eigenvalue corresponding to v. More precisely, show that if the coordinates of the "noise" $Cv - \lambda v$ are all smaller than $p/6$ in absolute value, then for any $\lambda' \neq \lambda$, $Cv - \lambda' v$ has at least one coefficient whose absolute value is greater than $p/6$ (hint: consider the last $(k+1)$ coordinates of $v = Gs$).

3. Decryption: to decrypt C, compute $Cv = CGs$ and find the value $\lambda \in \mathcal{M}$ such that $Cv - \lambda v$ has the smallest coefficients.

We have seen that as long as the coordinates of the noise remain smaller than $p/6$, there is a unique such approximate eigenvalue associated to λ, so the output of the decryption is correct. In practice $\mathcal{M}$ is small, so one can test all possible values of λ (see however exercice 6).

Exercise 6. Devise a decryption algorithm, that recovers λ knowing C and v (assuming that the noise is smaller than $p/6$).

4. Evaluation: if C encrypts λ and C' encrypts λ', then the homomorphic addition is the matrix addition $C + C'$, and the homomorphic multiplication is the matrix multiplication followed by flattening : $C'' = G^{-1}(CC'G)$.

(Note that addition can also be followed by flattening if one wants to keep the coefficients in $\{0,1\}$). Both operations will yield correct results if v is still an approximate eigenvector with the correct eigenvalue and a small enough noise. Let e, resp. e', be the noise in the encryption of λ, resp. λ'. If $C'' = C + C'$ (or $G^{-1}((C+C')G)$), then $C''v = \lambda v + e + \lambda' v + e' = (\lambda + \lambda')v + (e + e')$; thus C'' is a valid encryption of $\lambda + \lambda'$ and the size of the noise has increased by at most 1 bit. If $C'' = G^{-1}(CC'G)$, then

$$C''v = CC'v = C(\lambda' v + e') = \lambda'(\lambda v + e) + Ce' = \lambda'\lambda v + \lambda e + Ce'.$$

Since the coefficients of C are zeroes and ones, we see that the size of the resulting noise is $|\lambda e + Ce'|_2 \leq 1 + \max(|\lambda|_2 + |e|_2, |m|_2 + |e'|_2)$. If we assume an a priori bound B on the size of the plaintexts, this implies that the size of the noise grows linearly for additions as well as multiplication. This is much better than with the DGHV integer scheme, for which multiplications incurred a doubling of the size of the noise. In particular, the GSW scheme is still only somewhat (and not fully) homomorphic, but it can evaluate much more complicated functions.

There remains the issue of the growth of the plaintexts. Indeed, even if we only ever encrypt small plaintexts, there is no guarantee that they will remain small after several homomorphic evaluations. The workaround is to restrict homomorphic evaluations to functions for which given a bound on the size of the inputs, we have the same bound on the size of the output as well as a bound on the size of all auxiliary computed values. A special case is to restrict the message space to $\{0,1\}$ (i.e. a bitwise encryption scheme) and to evaluate homomorphically only the product (i.e. the logical AND) and the map $\lambda \mapsto 1-\lambda$ (i.e. the logical NOT). Since all boolean functions can be expressed with these two operations, this does not actually restrict the computing capabilities of the scheme.

Exercise 7. Order of operations, homomorphic computation of $m_1m_2m_3m_4$.

Exercise 8. Homomorphic multiplexing.

2.4 Other SHE schemes

To produce a somewhat homomorphically multiplicative scheme, it is easier to work in a (finite) ring R instead of $(\mathbb{Z}/p\mathbb{Z})^N$. The LWE problem can be transposed in the ring setting and is then called the Ring-Learning With Errors problem (R-LWE). To fix notation, we take $R = (\mathbb{Z}/p\mathbb{Z})[x]/(f(x))$, i.e. the ring of polynomials with coefficients in $\mathbb{Z}/p\mathbb{Z}$, modulo a polynomial f (which is usually of the form $x^{2^n}+1$). We are again given an "error" probability distribution χ, concentrated on polynomials whose coefficients are all small (in absolute value) compared to p. Let $s \in R$ be a secret element; we now consider couples of the form $(a_k, b_k) = (a_k, a_k s + e_k) \in R \times R$ where a_k is chosen uniformly randomly in R while e_k is sampled with respect to χ. The R-LWE problem is then:

Given many couples $(a_k, b_k) = (a_k, a_k s + e_k)$, recover s.

Similarly to the LWE problem, it is at least as hard as worst-case instances of lattices problems, but for ideal lattices. As such, it is expected to be easier than the LWE problem.

SHE schemes relying on the R-LWE problem are based on the following outline (proposed by Brakerski and Vaikunthanatan). The secret key is an element $s \in R = (\mathbb{Z}/p\mathbb{Z})[x]/(f(x))$. Here the message space, instead of being only $\{0,1\}$, now consists of polynomials of degree $(\deg f)-1$ with coefficients in $\{0,1\}$, i.e. we encode several bits at once (and the multiplication is given modulo 2 and modulo f). The ciphertexts are elements of $R[X]$, i.e. polynomials with coefficients in R.

1. Key generation: select a random element $s \in R$.
2. Encryption: to encrypt a message $m \in (\mathbb{Z}/2\mathbb{Z})[x]/(f(x))$, we choose a random element $a \in R$, a random small element $r \in R$ (following some error probability distribution) and output $P(X) = as + 2r + m - aX$. This is should be considered as being equivalent to the couple $(a, as+2r+m) \in R \times R$, which is close to what we have seen in the DGHV and Regev schemes.
3. Evaluation: addition and multiplication of ciphertexts are just the usual addition and multiplication of polynomials
4. Decryption: to decrypt a ciphertext $P(X) = a_0 + a_1X + \cdots + a_rX^r \in R[X]$, we compute $P(s) = \sum_i a_i s^i \in R$ and take its representative in $\mathbb{Z}[x]$ which has degree smaller than $\deg f$ and coefficients in $(-p/2, p/2)$, then reduce each coefficient modulo 2.

This scheme is indeed somewhat homomorphic: for each "fresh" ciphertext $P(X)$, its evaluation $P(s)$ is equal to the "noise" $2r+m$ which decrypts as m since r is small. Since the evaluation of polynomials is compatible with their addition and multiplication, we see that addition and multiplication of ciphertexts correspond to the addition and multiplication of the corresponding noises, which guarantees the correction of the decryption as long as the noises remain small.

Note that as in the integer scheme, it is mainly the multiplications that increase the noise. In this scheme, multiplications also have the drawback of increasing the length of the ciphertexts, i.e. the degree of the polynomials. In any case, this degree is bounded by the maximum of operations D this SHE scheme can support before ceasing to be homomorphic. It is also possible to keep the ciphertexts small (basically just couples of elements) by using a relinearization technique, but this incurs an expansion of the key size.

3 From somewhat homomorphic to fully homomorphic

What prevents a somewhat homomorphic encryption scheme to be fully homomorphic is the increase of the size of the noise as more operations are performed on the ciphertexts. Thus, we need to find a way to "refresh" the ciphertexts. The answer proposed by Gentry very simple yet intriguing: *we just have to evaluate the decryption function itself!*

More precisely, let (pk_1, sk_1) and (pk_2, sk_2) be two pairs of public/private key for a given somewhat homomorphic encryption scheme, and let $\overline{sk_1}$ be the encryption of (the bits of) sk_1 under pk_2. We then consider the *recryption* of a ciphertext c encoded under pk_1.

```
Recrypt($pk_2, \overline{sk_1}, c$) :
        for all $i$, $\overline{c}_i = \texttt{Encrypt}(pk_2, c_i)$ where $c_i$ is the $i$-th bit of $c$;
        output $c' = \texttt{Evaluate}(\texttt{Decrypt}, \overline{sk_1}, \overline{c}_1, \ldots, \overline{c}_m, pk_2)$.
```

Here `Decrypt` is considered as a boolean function (or a logical circuit) that takes as inputs the bits of the secret key and of the ciphertext. So we have added an outer layer of encryption with the second set of keys to the ciphertext c, then we have removed the inner encryption using the homomorphic evaluation of the decryption function. The result is an encryption of the plaintext for the second set of keys; we have thus eliminated the original noise (which is the purpose of `Decrypt`), but in doing so we may have generated a large new noise; this is ok as long as this new noise is smaller than the original one.

So all we need to construct a FHE scheme is a SHE scheme *that can handle its own decryption function.* In fact, we need a little more: it must be able to handle its decryption function and at least one more operation (otherwise this is useless). Such a SHE scheme is called *bootstrappable.* To evaluate a function in the corresponding FHE scheme, we just have to periodically "refresh" the ciphertexts by running the `Recrypt` procedure (and changing the key).

The periodic change of keys with this bootstrapping construction means that we need keys whose length is proportional to the complexity of the functions we evaluate (following Gentry, this is called a *leveled* FHE scheme). So we can opt instead for always keeping the same key, i.e. choosing $(pk_2, sk_2) = (pk_1, sk_1)$ in the `Recrypt` procedure. But in doing so we reveal $\overline{sk_1}$, which is an encryption of the private key with its own corresponding public key. It is a problem if this information can be used to attack the cryptosystem. Otherwise, i.e. if $\overline{sk_1}$ does not leak any useful information, the encryption scheme is called *circularly secure.* Actually, most known cryptosystems seem to satisfy this circular security assumption.

Remark. *The first step in the* `Recrypt` *procedure can usually be skipped. Indeed, it is not necessary that $\overline{c}_i$ is a safe encryption of c_i; all is needed is that $\overline{c}_i$ decrypts to c_i. But in most SHE constructions,* 0 *and* 1 *simply decrypt to* 0 *and* 1*, so it is possible to take $\overline{c}_i = c_i$.*

Bootstrapping can be rightfully considered as a rather contrived way to obtain fully homomorphic encryption, but up to now all “pure” FHE schemes rely on this process and on circular security. However, it is by far the main complexity bottleneck. A partial solution is batch bootstrapping: in some schemes (typically R-LWE based) it is possible to “pack” ciphertexts and operate on them simultaneously, in a SIMD fashion. The amortized cost of bootstrapping, or more generally any homomorphic operation, is thus much smaller than for a single message. Besides, if we consider leveled schemes, i.e. schemes in which the length of the key is proportional to the complexity of the functions to evaluate, then there exist algorithms for which bootstrapping is only optional. But of course, this requires to have an a priori bound on the function complexities.