Public-key encryption is a type of algorithm by which Alice can send Bob secret messages over public channels without them having exchanged any secret information in advance. The steps of public-key encryption are

1. **Key generation:** Bob runs a randomized key generation algorithm $Keys$ to produce a pair of keys: A secret key SK and a public key PK. He posts PK for anyone who is interested to use it.
2. **Encryption:** To communicate a message m, Alice runs an algorithm $Encrypt_{PK}(m)$ and sends its output c to Bob.
3. **Decryption:** Upon receiving ciphertext c, Bob runs $Decrypt_{SK}(c)$ to recover m.

An encryption scheme is *functional* if decryption inverts encryption: For every message m, and for every pair of keys (SK, PK) sampled by $Keys$, $Decrypt_{SK}(Encrypt_{PK}(m)) = m$. We also want the encryption scheme to be secure: An eavesdropper Eve who observes the whole interaction (she sees both the public key and the ciphertext) should not be able to infer anything about the message (see Figure 1 (a)).

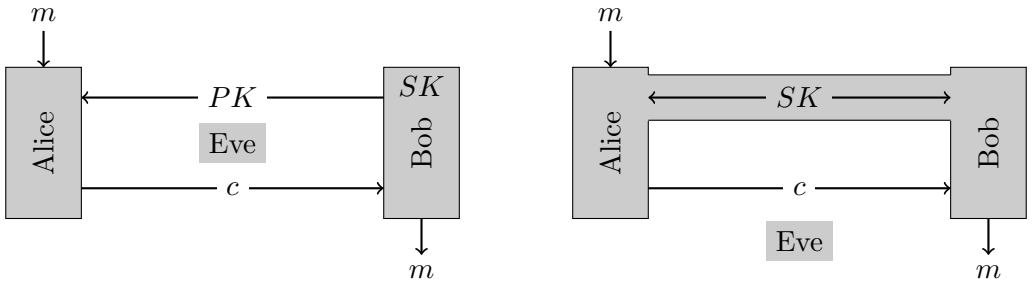

FIGURE 1: Public-key and private-key encryption. (a) In a public-key scheme Alice and Bob do not share any secrets prior to the interaction. Eve observes Bob's public key and Alice's ciphertext. (b) In a private-key scheme Alice and Bob have a shared secret key unknown to Eve. Eve observes Alice's ciphertext only.

Attaining functionality without security is easy: Alice sends the unaltered message m to Bob. Hiding m without requiring correctness is also easy: Alice doesn't send anything. What makes encryption interesting is that functionality and security should be achieved simultaneously.

1 The one-time pad

The *one-time pad* is an example of an encryption scheme that is simultaneously secure and correct. It is a *secret-key* scheme: Alice and Bob have to agree in advance on a (random) key SK that is unknown to anyone else (see Figure 1 (b)). The message m lives in $\mathbb{Z}_n = \{0, 1 \ldots, n-1\}$, i.e., the numbers modulo n. The secret key SK is a random number in $\mathbb{Z}_n$. Both encryption and decryption now use the shared secret key SK. They are given by

$$Encrypt_{SK}(m) = m + SK \bmod n \qquad \text{and} \qquad Decrypt_{SK}(c) = c - SK \bmod n$$

For instance, if $n = 42$ and the secret key is $SK = 25$, then $Encrypt_{SK}(34) = 17$ and $Decrypt_{SK}(17) = 34$.

The one-time pad is secure in the strongest possible sense. It hides not only the message m, but prevents external observer Eve of the ciphertext c in learning any partial information about the message m. The reason is that from Eve's perspective, ciphertexts arising from different messages are *identically distributed*. Just like in secure multiparty computation, this is the gold standard of security.

Definition 1. Random variables X and X' are *perfectly indistinguishable* if they take every value with the same probability, i.e., $\Pr(X = x) = \Pr(X' = x)$ for all possible outcomes x.

Definition 2. An encryption scheme is *perfectly secure* if for every pair of messages m and m', the ciphertexts $Encrypt_{SK}(m)$ and $Encrypt_{SK}(m')$ are perfectly indistinguishable.

Any piece of information about m ("$m = 10$ is even", "$m = 10$ is not prime") must make it distinct from some other m'. Perfect indistinguishability implies that the distinctness cannot be inferred by observing $Encrypt_{SK}(m)$. This is true even if Eve selects, and in particular knows m and m' in advance, and includes information that was not even envisioned when designing the encryption scheme.

For example, Eve may know that Bob is Alice's stockbroker, message 4 is an instruction to buy, message 11 is an instruction to sell, and one of those two messages is being communicated. Observing the ciphertext gives her no additional information which it was. In either case, the ciphertext in the one-time pad is equally likely to equal any of the numbers modulo 42.

The one-time pad is perfectly secure. If Eve observes ciphertext c, it is equally likely to have arisen from encrypting message m with secret key $c - m$ as it is to have arisen from encrypting m' with secret key $c - m'$ (see Figure 2). Both events have probability $1/n$. The random variables $Encrypt_{SK}(m)$ and $Encrypt_{SK}(m')$ are identically distributed.

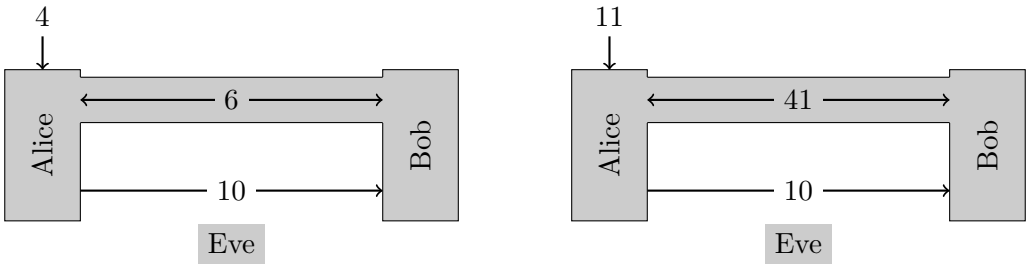

FIGURE 2: Indistinguishability of encryptions for the one-time pad. The modulus is $n = 42$. The ciphertext 10 is equally likely to have arisen from messages $m = 4$ and $m' = 11$. Events $c = 10, m = 4$ and $c = 10, m' = 11$ have the same probability $1/42$ as they each is consistent with a unique secret key.

A major shortcoming of the one-time pad is that the secret key has to be as long as the message. In modern settings where terabytes of information are regularly exchanged it is unrealistic to expect parties to coordinate terabyte-long secret keys in advance. Public-key encryption dispenses of this exchange of secret information altogether. Is there however *any* method of encryption that shares less secret information than the one-time pad? Shannon showed that the answer is no—as long as the standard is perfect security. To obtain succinct encryption we must lower the standard.

2 Security of encryption

One reasonable relaxation is to allow for a small amount of imperfection. A fair coin can in principle be distinguished from one in which the odds are 50.0001 heads and 49.9999 tails. In practice, it would take many coin flips to observe the difference.

Two random variables X and X' are ϵ-close if for every possible hypothesis H, the probability that H holds for X differs by at most ϵ from the probability that H holds for X' (in absolute value). Perfectly indistinguishable random variables are 0-close. It should stand to reason that there is not much harm in replacing this zero with a tiny ϵ.

In the case of the coin flip there are two possible hypothesis, heads or tails. The difference in probabilities of either hypothesis is 0.0001, so the outcomes of the two coins are 0.0001-close.

A ciphertext modulo $n = 42$ has 42 possible outcomes. ϵ-imperfect security says that the ciphertexts c and c' arising from say $m = 4$ and $m' = 11$ be ϵ-close. This means that the probability that the probabilities

that $c = 10$ and $c' = 10$, or any other value, differ by at most ϵ. It also means a lot more. The predicates "the ciphertext is 3 or 19", "the ciphertext is prime", "the ciphertext is odd but not 5 or 11" are all valid hypotheses. In an ϵ-imperfect encryption Eve cannot tell apart m and m' with advantage exceeding ϵ *no matter which hypothesis she makes.*

This security requirement is very strong. Unfortunately it is too strong to be useful. A 0.1-imperfect encryption, for instance, would still require the secret key to be as long as the message. To build succinct encryption we have to compromise more. How?

The key insight of modern cryptography is that *Eve is an algorithm.* Pragmatically, Eve can only test those hypotheses that she can code up and execute in a reasonable amount of time. When n is large this rules out a vast majority of hypotheses. The resulting notion of security is still meaningful, yet plausibly achievable.

Definition 3. Random variables X and X' are (t, ϵ)-computationally indistinguishable if for every algorithm H that can be coded up and executed in at most t steps (i.e, of complexity t), the probabilities that H accepts X and X' differ by at most ϵ.

An encryption scheme is computationally secure if for every pair of messages m and m', Eve's views when messages m and m' are encrypted are computationally indistinguishable. In public-key encryption, Eve's view consists of both the public key and the ciphertext.

Definition 4. A public-key encryption scheme is (t, ϵ)-computationally secure if for every pair of messages m and m', the joint random variables $(PK, Encrypt_{PK}(m))$ and $(PK, Encrypt_{PK}(m'))$ are (t, ϵ)-computationally indistinguishable.

A unique feature of cryptographic definitions like public-key encryption is that they posit both the existence and non-existence of algorithms. Functionality is about existence of encryption and decryption. Security is about non-existence of any efficient algorithm that can tell apart encryptions of two messages.

A central axiom of cryptography is that adversaries like Eve have significantly more resources at their disposal than honest parties like Alice and Bob. In a meaningful implementation, *Encrypt* and *Decrypt* may run in a few thousand steps, while (t, ϵ) security should hold for values of t and $1/\epsilon$ that should exceed any reasonable computational resources, typically on the order of 2^{100} or larger. As we will see, it is often useful to reason about these quantities asymptotically.

Unlike for secret-key encryption, perfectly secure public-key encryption schemes do not exist, regardless of key size. For Bob to be able to decrypt, for any two distinct messages m and m', the encryptions $Encrypt_{PK}(m)$ and $Encrypt_{PK}(m')$ can never be equal to one another, so the random variables $(PK, Encrypt_{PK}(m))$ and $(PK, Encrypt_{PK}(m'))$ have disjoint support. Random variables with disjoint support can be perfectly distinguished. In fact the decryption algorithm distinghishes them given access to the secret key. The challenge is to make them computationally indistinguishable without it.

To summarize, in a functional and secure public-key encryption, Eve's views $(PK, Encrypt_{PK}(m))$ and $(PK, Encrypt_{PK}(m'))$ (1) can be sampled efficiently by running *Encrypt* (2) are in principle distinguishable by *Decrypt* using knowledge of the secret key but (3) cannot be distinguished by any algorithm that can be coded up and executed in a reasonable amount of time. Towards building public-key encryption let's discuss a plausible example of efficiently sampleable random variables with all these properties.

3 The Decisional Diffie-Hellman assumption

The Decisional Diffie-Hellman (DDH) assumption has to do with exponentiation in finite groups. Let's start with the integers $\mathbb{Z}_p$ modulo p, where p is a very large prime number (think thousands of bits). Recall that we can do not only addition and subtraction, but also multiplication and division modulo p, as long as we do not divide by zero. If we exclude zero, we obtain the multiplicative group $\mathbb{Z}_p^*$ of size $p-1$ in which multiplication and division of any two numbers is well-defined.

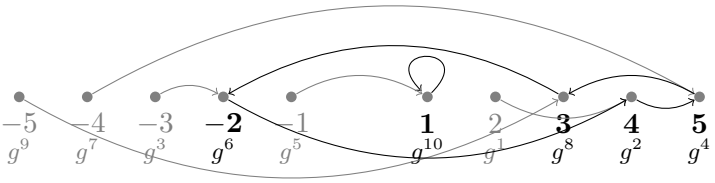

FIGURE 3: Powering in $\mathbb{Z}_{11}^*$. Arrows indicate squaring. The squares (in boldface) form the subgroup $\mathbb{G}$ of quadratic residues modulo 11. Each has one positive and one negative square root.

Let's talk about exponentiation. Given a group element g and an integer x, the power g^x can be calculated by multiplying g with itself x times. We'll be interested in the setting where x itself is very large, on the order of p itself. There is then a much faster way to exponentiate: If x is even, first calculate g^2 then raise it to the $(x/2)$-th power recursively. If x is odd, do the same with $x-1$ and multiply the result by g. This fast exponentiation algorithm reduces the number of multiplications from x to $O(\log x)$, so the whole procedure can be implemented in time roughly quadratic in the logarithm of p, namely the number of bits it takes to write down a number in $\mathbb{Z}_p^*$.

The inverse of exponentiation is taking discrete logarithms: Given g and h, find x so that g^x equals h (if one exists). How difficult is this? While this is not known, the complexity of the best known algorithms is governed by the largest prime factor of $p-1$. So if $p-1$ were prime, finding discrete logarithms in $\mathbb{Z}_p^*$ might be exponentially harder than exponentiating.

Unfortunately p and $p-1$ cannot both be prime. (Don't ask the obvious question.) The next best thing is to have $p-1$ equal twice some prime number q. Such prime numbers p are called *safe primes*. Mathematicians believe but cannot prove that there are infinitely many of them. In any case, there are explicit safe primes of the right order of magnitude for encryption.

The Decisional Diffie-Hellman assumption postulates that discrete logarithms are not only difficult to find, but they are indistinguishable from random numbers in the following sense. Let g be an element of $\mathbb{Z}_p^*$. We'll assume that p, q, and g are publicly known.

DDH assumption for base g: Random variables (g^X, g^Y, g^{XY}) and (g^X, g^Y, g^Z) are (t, ε)-computationally indistinguishable, where X, Y, Z are random integers modulo $p-1$.

The reason that X, Y, and Z are taken modulo $p-1$ is that exponentiation wraps around modulo the order of the group, i.e., $h^{p-1} = 1$ for all elements h of $\mathbb{Z}_p^*$.

If discrete logarithms were easy to calculate then the DDH assumption couldn't be true. A distinguisher could recover X and Y from g^X and g^Y and use this to distinguish g^{XY} from the random group element g^Z. On its own, hardness of discrete logarithms is insufficient for DDH. There are choices for g for which DDH does not hold but taking discrete logarithms might still be difficult.

To explain how these arise let's take $p = 11$, which is a safe prime, and $g = 2$ (see Figure 3). The chances that g^{XY} is an even power of g is $3/4$; this happens when at least one of X and Y is even. But g^Z is an even power of g only with probability $1/2$. So a distinguisher that can tell apart even and odd powers of g breaks the DDH assumption. This can be done efficiently: h is an even power of g if and only if $h^5 = 1$.

This strategy works for any safe prime p as long as g generates the group $\mathbb{Z}_p^*$, namely the powers $g, g^2, \ldots, g^{p-1}$ cover the whole group: To distinguish $h = g^{XY}$ from $h = g^Z$, check if $h^{(p-1)/2} = 1$.

Had we chosen $g = 4$ instead of $g = 2$ the distinguisher wouldn't have worked. In this case, g does not generate the whole group $\mathbb{Z}_p^*$, but only its subgroup $\mathbb{G} = \{1, 3, 4, 5, -2\}$ of *quadratic residues* modulo 11, namely the even powers of the old g. Now g^{XY} and g^Z are uniformly random in $\mathbb{G}$, and the DDH

assumption says that (g^X, g^Y, g^{XY}) are indistinguishable from three independent random elements of $\mathbb{G}$.[1]

The DDH assumption is believed to be plausible whenever g is a quadratic residue (except 1), that is the square of a number, modulo a safe prime p. The best known attacks have complexity exponential in the cube root of p, so DDH might hold with parameters $t/\varepsilon \approx \exp(\sqrt[3]{p})$.[2]

4 Diffie-Hellman key exchange and El Gamal encryption

The DDH assumption gives an easy mechanism by which Alice and Bob can securely agree on a private key without prior interaction. This is the beautiful Diffie-Hellman key exchange:

Algorithm *DH* (Diffie-Hellman key exchange)

1 Bob chooses a random X.
Alice chooses a random Y.
2 Bob sends g^X to Alice.
Alice sends g^Y to Bob.
3 Bob raises Alice's message to the Xth power.
Alice raises Bob's message to the Yth power.

Alice and Bob both obtain g^{XY}, which they declare to be their secret key. What Eve sees from this interaction are the values g^X and g^Y. The DDH assumption says that she gains no information about the secret key g^{XY} because relative to g^X and g^Y, g^{XY} can be simulated by an independent random group element g^Z.

Once we have a key exchange mechanism it is not difficult to obtain public-key encryption. Suppose Bob sends his message g^X before Alice does. Then Alice knows the secret key g^{XY} and can use it as a one-time pad. Here is the resulting public-key encryption.

Algorithm *ElGamal* (El Gamal encryption)
Key generation: A random X is Bob's secret key. $PK = g^X$ is his public key.
Encryption: Message m is an element of $\mathbb{G}$. Alice encrypts it as $(g^Y, PK^Y \cdot m)$ with random Y.
Decryption: To decrypt (h, c), Bob outputs $h^{-X} \cdot c$.

The scheme is functional. The decryption of $(g^Y, (g^X)^Y \cdot m)$ equals $(g^Y)^{-X} \cdot g^{XY} \cdot m = m$. It is also secure under the DDH assumption:

Proposition 5. *If the (t, ε)-DDH assumption holds then El Gamal encryption is $(t - t_\times, 2\varepsilon)$-computationally secure where $t_\times$ is the cost of group multiplication by a fixed element.*

Proof. These proofs always go by contrapositive. Suppose Eve can 2ε-distinguish her views when m and m' are encrypted with program P. Then one of the two views, say $(PK, Encrypt_{PK}(m))$, is ε-distinguished from the model random variable (g^X, g^Y, g^Z) by the same P. Had this been false, both $(PK, Encrypt_{PK}(m))$ and $(PK, Encrypt_{PK}(m'))$ would have been ε-indistinguishable from (g^X, g^Y, g^Z) they would be mutually 2ε-indistinguishable.

As g^Z is a random group element independent of (g^X, g^Y), (g^Y, g^X, g^Z) is identically distributed to $(g^X, g^Y, g^Z \cdot m)$. Unwinding *Encrypt* we see that P ε-distinguishes $(g^X, g^Y, g^{XY} \cdot m)$ from $(g^X, g^Y, g^Z \cdot m)$. Then the program P' that, on input (h, i, j) outputs $P(h, i, j \cdot m^{-1})$ "cancels out" m and ε-distinguishes (g^X, g^Y, g^{XY}) from (g^X, g^Y, g^Z). P' does the same work as P plus a multiplication by m^{-1}. If P has complexity less than $t - t_\times$ than P' has complexity less than t. Therefore P' breaks the (t, ε)-DDH assumption. □

[1] X, Y, and Z can now be taken modulo q instead of $p - 1 = 2q$ because g^X etc. still cover all of $\mathbb{G}$ uniformly.
[2] There are other so-called elliptic curve groups in which DDH is thought to be even more secure.

Unlike for the one-time pad, the encryption algorithm here is randomized. Different runs on the same message will produce different ciphertexts. Randomness is essential for secure public-key encryption. If encryption were deterministic, encryptions of m and m' would be easily distinguishable. Can you see why?

One important consequence of Definition 4 is that it is secure to encrypt multiple messages under the same public key. Even if Alice happens to El Gamal encrypt the same message twice, Eve will have no idea that this happened. The two encryptions are "computationally" independent given Bob's public key.

One annoyance of El Gamal encryption is that the message space is the group of quadratic residues $\mathbb{G}$, which is not nicely structured. It would be more convenient if the messages were say integers between 1 and q. Fortunately there is an efficient conversion between the two representations

$$\mathbb{G} \leftrightarrow \{1, \ldots, q\}.$$

To convert an integer into a quadratic residue, square it; to convert a quadratic residue h into an integer, calculate $h^{(q+1)/2}$ modulo p as an integer between $-q$ and q and take its absolute value. This works because $h^{(q+1)/2}$ and $-h^{(q+1)/2}$ both square to $h^{q+1} = h$, so the positive one must be equal to the integer representation of h. (See Figure 3. The elements of $\mathbb{G}$ are exactly the squares of the numbers 1 to 5. To go in the other direction, raise to the third power and take the absolute value.)

5 Learning with Errors

The Learning with Errors (LWE) assumption is about the hardness of solving noisy linear equations in rings. The domain is the ring $\mathbb{Z}_q$ of integers modulo q. There are several variants of this assumption which turn out to be roughly the same. We will work with a variant that is useful for building public-key encryption.

A natural way to create a *solvable* system of n linear equations in n variables modulo q is to plant a secret solution $\mathbf{x}$ and choose the system at random in a way that is consistent with this solution. For example, if $n = 3$ and $q = 191$ the system may look like this:

$$\begin{aligned} 143x_1 + 156x_2 + 161x_3 &= 153 \\ 156x_1 + 30x_2 + 33x_3 &= 188 \\ 150x_1 + 24x_2 + 154x_3 &= 87 \end{aligned}$$

The "secret" $\mathbf{x}$ can be found via modular Gaussian elimination. In this example the solution is $\mathbf{x} = (3, -1, 4)$. We will insist that the solution $\mathbf{x}$ is *b-short*: The numbers $x_1, \ldots, x_n$ are between $-b$ and b modulo q, with b much smaller than $q/2$. In our example $b = 5$. Things change dramatically if noise values in the range $-b$ to b are added to the right-hand side:

$$\begin{aligned} 143x_1 + 156x_2 + 161x_3 &= 151 \pm 5 \\ 156x_1 + 30x_2 + 33x_3 &= 188 \pm 5 \\ 215x_1 + 138x_2 + 49x_3 &= 92 \pm 5. \end{aligned}$$

A solution $\mathbf{x}$ can still be found by applying Gaussian elimination to the system with the corrupted right-hand side. In this example, if we ignore the ± 5 on the right, the unique solution is $(59, 41, 31)$. This solution is not short. Finding a solution that is short seems to be much more difficult. We know that one must exist because it was planted in there.

There are two obvious strategies. The first one is to try all possible candidate solutions $\mathbf{x}$. As each x_i can take any value between $-b$ and b, there are $(2b+1)^n$ solutions to try. Another strategy is to try all possible noise cancellations. Each noise entry can take values between $-b$ to b so there are again $(2b+1)^n$ noise patterns $\mathbf{e}$ to try. (Once the noise pattern has been guessed the short $\mathbf{x}$ is recovered using Gaussian elimination.) Both these algorithms have complexity about $(2b+1)^n$, which is impractical when n is in the

hundreds. There is a third strategy that is very efficient but only when q is very large (about $b \cdot 2^{n/\log n}$). As long as n is moderately large and q is not too large, recovering $\mathbf{x}$ looks like a very hard problem.

Search-LWE assumption for short secrets and n samples**:** There is no program of complexity t that, given a $n \times n$ random matrix A with entries modulo q and the n-dimensional vector $A\mathbf{x}+\mathbf{e}$ as inputs, finds $\mathbf{x}$ (with high probability), where both $\mathbf{x}$ and $\mathbf{e}$ are random b-short n-dimensional vectors modulo q.

Unless q is very large, this assumption is plausible with the complexity t set to about $O(b)^n$.

Intuitively, the search-LWE assumption says that in modular arithmetic, it is hard to solve an *almost*-solvable system of linear equations. In ordinary arithmetic with real numbers this is false because algorithms like gradient descent can find excellent approximate solutions. Such algorithms do not work in modular arithmetic. The only currently available alternative is Gaussian Elimination, which is not robust to the corruptions.

Just like in the case of DDH, for designing public-key encryption a decisional assumption is more useful. The (decisional) LWE assumption states that the right-hand side of the noisy system of equations is indistinguishable from random.

Learning with errors (LWE) assumption: The random variables $(A, A\mathbf{x}+\mathbf{e})$ and $(A, \mathbf{r})$, where $\mathbf{r}$ is a random n-dimensional vector modulo q, independent of A, are (t, ε)-computationally indistinguishable.

Unlike in the case of discrete logarithms, the $O(b)^n$-search and $(O(b)^n, \Omega(b)^{-n})$ decision LWE assumptions are essentially equivalent: The problems of finding a solution and distinguishing an almost-solvable system from a random one are roughly equivalent in computational complexity.

6 LWE-based key exchange

There are several algorithms for public-key encryption whose security is based on the LWE assumption. We will describe an algorithm for key exchange of Ding, Xie, and Lin that is not the most versatile one but I find easiest to understand. Just like Diffie-Hellman key exchange led to El Gamal encryption, this algorithm can be converted into an LWE-based public-key encryption scheme.

Algorithm DXL (Ding-Xie-Lin approximate key exchange)

1 Alice samples a random b-short row vector $\mathbf{x}'$.
Bob samples a random b-short column vector $\mathbf{x}$.
2 Alice and Bob sample a uniformly random n by n matrix A.
3 Bob sends $A\mathbf{x}+\mathbf{e}$ to Alice, with a random b-short $\mathbf{e}$.
Alice sends $\mathbf{e}'+\mathbf{x}'A$ to Bob, with a random b-short $\mathbf{e}'$.
4 Alice multiplies Bob's message by $\mathbf{x}'$ on the left.
Bob multiplies Alice's message by $\mathbf{x}$ on the left.

Alice obtains the value $\mathbf{x}'(A\mathbf{x}+\mathbf{e}) = \mathbf{x}'A\mathbf{x}+\mathbf{x}'\mathbf{e}$. Bob obtains the value $(\mathbf{e}'+\mathbf{x}'A)\mathbf{x} = \mathbf{e}'\mathbf{x}+\mathbf{x}'A\mathbf{x}$. As A is a random matrix, both of them are very close to uniformly random numbers modulo q.[3] But they are not exactly the same.

This is where the LWE assumption kicks in. As $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{e}$, and $\mathbf{e}'$ are b-bounded, the dot products $\mathbf{x}'\mathbf{e}$ and $\mathbf{e}'\mathbf{x}$ have magnitude at most b^2n. If q is sufficiently larger than b^2n, Alice's and Bob's keys are still very close to one another relative to their magnitude:

$$\mathbf{x}'(A\mathbf{x}+\mathbf{e}) \approx \mathbf{x}'A\mathbf{x} \approx (\mathbf{e}'+\mathbf{x}'A)\mathbf{x}.$$

The small discrepancy in Alice's and Bob's keys does not affect the utility of key exchange.

[3] Some error is incurred by the event $\mathbf{x}=0$ or $\mathbf{x}'=0$, but this is extremely rare.

How about security? Eve observes the messages $(A, A\mathbf{x}+\mathbf{e}, \mathbf{e}'+\mathbf{x}'A)$. Her public key is $\mathbf{x}'(A\mathbf{x}+\mathbf{e})$. Security for key exchange requires that the messages and Alice's key[4] should be computationally indistinguishable from a pair of independent random variables. By the LWE assumption, $(A, A\mathbf{x}+\mathbf{e}, \mathbf{e}'+\mathbf{x}'A, \mathbf{x}'(A\mathbf{x}+\mathbf{e}))$ is indistinguishable from $(A, \mathbf{r}, \mathbf{e}'+\mathbf{x}'A, \mathbf{x}'\mathbf{r})$ for a random $\mathbf{r}$.

We would now like to argue that the last random variable is indistinguishable from a sequence of independent random numbers modulo q, but it is not clear how to do that. One way to proceed is to modify the key exchange so that Alice adds a little bit of extra noise to her key.

4 Alice multiplies Bob's message by $\mathbf{x}'$ on the left and adds a random b-short number e'' to it.

This modification is sufficient to complete the security proof for DXL key exchange.

Theorem 6. *Assuming (t, ε)-LWE for $n+1$ equations, the messages exchanged by DXL together with Alice's public key are $(t - t_\sim, 2\varepsilon)$-computationally indistinguishable from independent random strings over $\mathbb{Z}_q$, where $t_\sim$ is the time it takes to sample $A\mathbf{x}+\mathbf{e}$ on input A.*

Proof sketch. Eve's view is the random variable

$$(A, A\mathbf{x}+\mathbf{e}, \mathbf{e}'+\mathbf{x}'A, e''+\mathbf{x}'(A\mathbf{x}+\mathbf{e})). \tag{1}$$

Random variable (1) is $(t - t_\sim, \varepsilon)$-indistinguishable from

$$(A, \mathbf{r}, \mathbf{e}'+\mathbf{x}'A, e''+\mathbf{x}'\mathbf{r}) \tag{2}$$

assuming (t, ε)-LWE, for a distinguisher between these two can be turned into one for LWE at an additional cost of $t_\sim$ (the time it takes to sample the extra input $\mathbf{e}'+\mathbf{x}'A$). If we combine A and $\mathbf{r}$ into a single matrix $[A|\mathbf{r}]$, (2) can be viewed as an LWE instance with $n+1$ equations as columns $([A|\mathbf{r}], [\mathbf{e}'|e''] + \mathbf{x}'[A|\mathbf{r}])$. It is therefore $(t - t_\sim, \varepsilon)$-indistinguishable from four independent random strings

$$(A, \mathbf{r}, \mathbf{r}', \mathbf{r}''). \tag{3}$$

modulo q. As (1) and (2) are $(t - t_\sim, \varepsilon)$-indistingishable and (2) and (3) are $(t - t_\sim, \varepsilon)$-indistinguishable, (1) and (3) must be $(t - t_\sim, 2\varepsilon)$-indistinguishable. □

7 Encryption and computation

Humans have been trying to hide information since ancient times. The Caesar cipher, attributed to Roman emperor Julius Caesar, encrypts a message by shifting each letter three symbols ahead, e.g., `apple` would encrypt to `dssoh`. By modern standards Caesar's cipher is hopelessly easy to break. More sophisticated encryption schemes were proposed later. In World War II the Germans encrypted their communications using the elaborate Enigma machine. The Enigma cipher was famously cracked by Polish and British mathematicians supporting the Allied troops.

Modern insights about computation explain why such attempts at encryption failed. One such insight is the universality of computation. It is impossible to hide an algorithm, including the ones for encryption and decryption, because one can search through all algorithms. Hiding information cannot rely on the secrecy of the algorithms. It has to make explicit use of randomness. The one-time pad clearly embodies this principle. Knowing how it works helps you not one bit in figuring out the message.

To improve on the one-time pad one must first understand its weakness. Shannon articulated it well: The secret key must be at least as long as the message as long as *perfect* security is the standard. A natural idea to overcome this limitation is to replace the random key in the one-time pad by the output of what is now called a *pseudorandom generator*. This is a deterministic algorithm that converts a short truly

[4] Owing to the small disagreement Alice's and Bob's key are not identical.

random seed into a potentially infinite stream of "random-looking" bits. The output of a pseudorandom generator is a proxy for an endless one-time pad.

In the 1960s and 1970s cryptographers took up the task of designing such pseudorandom generators, coming up with ingenious proposals like DES. They were aware that their designs could be in principle broken using brute force given unreasonably large computational power. How could they be sure that there was no better, faster way to attack?

The answer is that they didn't and we still don't. It is possible that all encryption methods that improve on the one-time pad are insecure. The best we can say about their security in this day in age is that many clever people have tried to break them, and all of them have failed so far.

A major insight of cryptographers in the 1980s was that even though encryption schemes cannot be proved unbreakable, the assumptions that underlie their security can be precisely stated. The DDH and LWE assumptions are a case in point. Once we accept DDH, we know that El Gamal encryption must be secure. To gain confidence in DDH, we can test all Great Algorithms we know on it. Watching them all fail should increase our confidence in its veracity.

Public-key cryptography was proposed in a paper of Diffie and Hellman in 1976. The promise of secret communication between two people who have never met was revolutionary. Their work was not motivated by practical needs. Wide-area networks like the internet had to wait for another fifteen years. Once long-distance communication over infrastructure beyond anyone's control became a reality, public-key encryption proved indispensable. It is difficult to imagine how e-commerce would operate without it.

In 1994 Peter Shor demonstrated that discrete logarithms can be found quickly on a quantum computer. If a quantum computer is built the security of Diffie-Hellman key exchange and El Gamal encryption will be compromised. The alternative RSA encryption whose security rests on the inability to factor large numbers would also be broken.

These discoveries motivated the search for encryption schemes that are impervious to potential quantum attacks. A line of seemingly unrelated works pioneered by Ajtai in 1996 led to the LWE assumption. LWE was formulated by Regev in 2006. He also designed an encryption scheme based on it. So far LWE appears resilient to quantum computers.

LWE opened up new possibilities that were hardly contemplated before. Starting with work of Gentry in 2009 we now have fully homomorphic encryption schemes that are based on LWE-like assumptions. In such a scheme encryptions of two messages can be turned into encryptions of their sums and products without knowledge of the secret key. As any circuit can be built out of plus and times gates, in principle fully homomorphic encryption can perform arbitarary computations on encrypted data.

As all public-key encryption schemes rest on unproven computational hardness assumptions, there is always the danger that the discovery of Great new Algorithms will invalidate their security. The machine learning revolution of the last decade has given us new algorithms with unmatched capabilities. So far these algorithms haven't had much success in breaking encryption. Will future advances in machine learning incapacitate encryption, or will they remain ineffective? Computer scientists have yet to answer this fascinating question.