If our finite field is $\mathbf{F}_{p^f}$, we first choose an $\mathbf{F}_p$-basis of this field, so that every element corresponds to an f-tuple of elements of $\mathbf{F}_p$; then such an f-tuple gives an integer less than p^f if we consider the coordinates as digits of an integer written to the base p. **Warning:** This gives a 1-to-1 correspondence between $\mathbf{F}_{p^f}$ and $\mathbf{Z}/p^f\mathbf{Z} = \{0, 1, 2, \ldots, p^f - 1\}$. But these two sets have a very different structure under addition and multiplication. The first is a *field*, i.e., all of the $p^f - 1$ nonzero elements have inverses, while the second is a *ring* in which p^{f-1} of the p^f elements (the multiples of p) fail to have inverses.

We now describe the Diffie–Hellman method for generating a random element of a large finite field $\mathbf{F}_q$. We suppose that q is public knowledge: everyone knows what finite field our key will be in. We also suppose that g is some fixed element of $\mathbf{F}_q$, which is also not kept secret. Ideally, g should be a generator of $\mathbf{F}_q^*$; however, this is not absolutely necessary. The method described below for generating a key will lead only to elements of $\mathbf{F}_q$ which are powers of g; thus, if we really want our random element of $\mathbf{F}_q^*$ to have a chance of being any element, g must be a generator.

Suppose that two users A (Aïda) and B (Bernardo) want to agree upon a key — a random element of $\mathbf{F}_q^*$ — which they will use to encrypt their subsequent messages to one another. Aïda chooses a random integer a between 1 and $q - 1$, which she keeps secret, and computes $g^a \in \mathbf{F}_q$, which she makes public. Bernardo does the same: he chooses a random b and makes public g^b. The secret key they use is then g^{ab}. Both users can compute this key. For example, Aïda knows g^b (which is public knowledge) and her own secret a. However, a third party knows only g^a and g^b. If the following assumption holds for the multiplicative group $\mathbf{F}_q^*$, then an unauthorized third party will be unable to determine the key.

Diffie–Hellman assumption. It is computationally infeasible to compute g^{ab} knowing only g^a and g^b.

The Diffie–Hellman assumption is *a priori* at least as strong as the assumption that discrete logarithms cannot be feasibly computed in the group. That is, if discrete logarithms can be computed, then obviously the Diffie–Hellman assumption fails. Some people would conjecture that the converse implication also holds, but that is still an open question. In other words, no one can imagine a way of passing from g^a and g^b to g^{ab} without first being able to determine a or b; but it is conceivable that such a way might exist.

Example 3. Suppose we're using a shift encryption of single–letter message units in the 26-letter alphabet (see Example 1 of §III.1): $C \equiv P + B \ mod\ 26$. (We're using B rather than b to denote the shift key so as not to confuse it with the b in the last paragraph.) To choose B, take the least nonnegative residue modulo 26 of a random element in $\mathbf{F}_{53}$. Let $g = 2$ (which is a generator of $\mathbf{F}_{53}$). Suppose Aïda picked at random $a = 29$, and looked up Bernardo's public 2^b, which is, say, $12 \in \mathbf{F}_{53}$. She then knows that the enciphering key is $12^{29} = 21 \in \mathbf{F}_{53}$, i.e., $B = 21$. Meanwhile, she