GUESS AGAIN: UNCONDITIONALLY SECURE PUBLIC-KEY ENCRYPTION (WITH POSSIBLE DECRYPTION ERRORS)

Algorithm Specifications and Performance

1. THE PROTOCOL

The public-key protocol in this section is for transmitting a single secret bit from Alice (the sender) to Bob (the receiver), but first we describe the offline phase where Alice builds a library of random walks that she will re-use to transmit multiple bits.

1.1. **Pre-computation, the offline phase.** Alice pre-computes a large (L entries) private library of random walks starting at integer points a_i from the interval $[0, n-1]$. More specifically, Alice builds two (sub)libraries, with $L/2$ entries in each: in one sublibrary, there are random walks with $f(n)$ steps, while in the other one there are random walks with $g(n)$ steps. To build either sublibrary, Alice selects, uniformly at random on integers from the interval $[0, n-1]$, a starting point a of her random walk. She then does a random walk starting at the point a with the number of steps corresponding to the sublibrary she is building. She eventually creates a library of L pairs (a_i, A_i), where a_i is the starting point of a random walk, and A_i is the end point. Half of these walks correspond to random walks with $f(n)$ steps, the other half to random walks with $g(n)$ steps. Some of the pairs may be repeated, but they still count toward the total of L entries in the library.

Finally, Alice moves each end point $A = A_i$ either $\frac{1}{2}$ left or $\frac{1}{2}$ right, with probability $\frac{1}{2}$. She then moves the corresponding a_i $\frac{1}{2}$ in the same direction. (This is needed to avoid situations where $a = b$ or $A = B$.)

1.2. **Protocol description.** The protocol description below includes parameters $n, f(n), g(n), h(n), L, R$ that are specified in Section 1.3 below. This protocol is for transmitting a single secret bit from Alice to Bob. We note though that in the current implementation, Bob's public key generated at Step 0 is used for encrypting multiple bits.

0. **Key generation.** Bob does a number of random walks with $h(n)$ steps, starting at points b_i uniformly distributed on integers from the interval $[0, n-1]$, with end points B_i, until he gets R random walks with $B_i < n-1$. The (ordered) tuple of R points $B_i < n-1$ is Bob's public key. (Note that some of the B_i can be repeated in this tuple.)
   Bob publishes his ordered tuple of R points B_i.

   Steps 1 through 4 are a loop that is repeated R times, by the number of published points B_i. Alice works with one point $B = B_i$ at a time.

1. Alice chooses, with probability $\frac{1}{2}$, between $f(n)$ and $g(n)$ steps, and then, again with probability $\frac{1}{2}$, between the two conditions: (1) $A < B$ and (2) $B < A$.
2. Alice selects, from her pre-computed library of random walks, among all random walks satisfying the conditions she chose at Step 1 and the condition $a > B$, one random walk uniformly at random. If there are no random walks satisfying these conditions in her library, Alice goes back to Step 0 and grows her library by adding more random walks satisfying $a > B$, for this particular B.
3. Let a_0 be the starting point of the random walk selected at Step 2. If the random walk selected by Alice at Step 3 has $f(n)$ steps and satisfies $A < B$, she chooses the interval $\{x < a_0\}$. If it has $g(n)$ steps and satisfies $A < B$, she chooses the interval $\{x > a_0\}$. If it has $f(n)$ steps and satisfies $A > B$, she chooses the interval $\{x > a_0\}$. If it has $g(n)$ steps and satisfies $A > B$, she chooses the interval $\{x < a_0\}$.
4. Alice assumes that Bob's decryption key b corresponding to the current point B is in the interval she selected at Step 4 of the protocol and encrypts her bit accordingly, i.e., by labeling the selected interval with her secret bit c and the other interval with the bit $1-c$. She then keeps the label of the interval $\{x < a_0\}$, to send it to Bob at Step 6.

   This completes the loop.

5. Alice sends to Bob an ordered tuple of R points a_0 (in the order corresponding to the ordered tuple of R points B_i), together with the corresponding labels of the intervals $\{x < a_0\}$. It is assumed that, for each particular a_0, if the interval $\{x < a_0\}$ is labeled by a bit c, then the interval $\{x > a_0\}$ is labeled by the bit $1-c$. Thus, the ciphertext consists of an ordered tuple of R points a_i and labeling of each interval $\{x < a_i\}$ by a bit.
6. For each point a_i Bob received from Alice at Step 6, there is the corresponding public point B_i. Bob then takes his private b_i corresponding to that B_i and recovers the bit corresponding to the label of the interval $\{x < a_i\}$ or $\{x > a_i\}$ where his b_i is. Having recovered R bits like that, Bob selects the bit that has been recovered more times than the other bit. This is the result of his decryption.

1.3. **Parameter values.** Suggested parameter values are: $n = 256$, $h(n) = 2000$, $g(n) = 2000$, $f(n) = 120,000$, $R = 2000$, $L = 2^{20}$.

1.4. **Private keys.** Below we summarize public as well as private information relevant to this protocol.

Alice's private key: points A_i (the end points of Alice's random walks).

Bob's private key: an ordered set of R points b_i (the starting points of Bob's random walks).

1.5. **Public information.** Public information consists of:

Public parameters: $n, h(n), f(n), g(n), L, R$.

Transmitted information: an ordered set of R points a_0.

Bob's public key: an ordered set of R points B_i (the end points of Bob's random walks).

1.6. **Keys size.** With suggested parameters, Bob's public key consists of $R = 2000$ points B_i. With $n = 256$ and $h(n) = 2000$, the typical range for B_i is $[-500, 255]$, which means the size of Bob's public key is going to be on the order of $9 \cdot 2,000 = 18,000$ bits.

Bob's private key is an ordered set of R points b_i corresponding to the R public values of B_i. Since b_i is in the interval $[0, 255]$, the size of Bob's private key with suggested parameters is on the order of $8 \cdot 2,000 = 16,000$ bits.

1.7. **Ciphertext size.** Alice's ciphertext (for encrypting a single bit) consists of R instances of:
(a) a point a_0;
(b) labeling of the interval $\{x < a_0\}$ with a bit.

Since the typical size of a is 8 bits, the total size of Alice's ciphertext for encrypting a single bit (with suggested parameters) is about $(8+1) \cdot 2,000 = 18,000$ bits. This is a pretty large expansion factor, but this is the price to pay for security against a computationally unbounded adversary.

1.8. **Security definition.** Our encryption scheme is IND-CCA2 secure, see Proposition 3 in the proposal narrative.

1.9. **Security strength category.** Security strength category: 5 in the classification from CFP 4.A.5.

However, security of our scheme is actually much stronger than that since we claim security against computationally unbounded (passive) adversary, which is possible since we allow decryption errors with some controlled probability, see Section 1.13 below.

1.10. **Reference implementation and optimized implementation.** We provide a reference implementation and an optimized implementation, as requested in the Call for Proposals. The optimized implementation is more efficient due to an optimized structure of the precomputed library of random walks.

1.11. **Performance.** With specified parameters, in our optimized implementation the online phase of encrypting a single bit takes about 0.016 sec on a ThinkPad Lenovo T460 computer with the Intel Core i7-6600U CPU @ 2.60GHz $\times$ 4 processor and 12GB memory, without using parallelization. Encrypting 32 bytes takes about 4 sec, again without parallelization.

1.12. **Memory requirements.** Memory required to run the protocol (including the offline phase) is 32MB.

1.13. **Probability of correct decryption.** Probability of correct decryption (by the legitimate party) of a single bit is 0.999996. Thus, if this protocol is used for KEM to establish a shared 256-bit secret key, the probability of successful key establishment will be $(0.999996)^{256} \approx 0.99898$.

This probability can be increased by increasing the parameter R (the number of published points B_i), but then encryption will take more time, so there is a trade-off between efficiency of encryption and correctness of decryption.