Name of the proposed cryptosystem: Guess Again

Principal submitter: Vladimir Shpilrain, shpil@groups.sci.ccny.cuny.edu, +1(917)860-6572, Department of Mathematics, The City College of New York, 161 Convent Avenue, New York, NY 10031

Auxiliary submitters: Mariya Bessonov, Alexey Gribov, and Dima Grigoriev

Inventors: Mariya Bessonov, Dima Grigoriev, and Vladimir Shpilrain

Developer: Alexey Gribov

Owners: Mariya Bessonov, Alexey Gribov, Dima Grigoriev, and Vladimir Shpilrain

Signature of the submitter: Vladimir Shpilrain

Backup point of contact: Alexey Gribov, gribov.alesha@gmail.com, +1(914)572-4923, 174 Pinewood Road, Apt.64, Hartsdale, NY 10530

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

ABSTRACT. We offer a public-key encryption protocol where decryption of a single bit by a legitimate party is correct with probability p that is greater than 1/2 but less than 1. At the same time, a computationally unbounded (passive) adversary correctly recovers the transmitted bit with probability exactly 1/2. By using known tools, the probability for a legitimate party to decrypt the bit correctly can then be amplified to be arbitrarily close to 1, while for a computationally unbounded passive adversary this probability is still 1/2.

1. PREFACE

It is well known (and easy to show) that unconditionally secure (i.e., secure without any computational assumptions) public-key encryption is impossible if the legitimate receiver decrypts correctly with probability exactly 1. The question is: what if this probability is less than 1? More precisely, what if the sender transmits a single encrypted bit and the legitimate receiver decrypts it correctly with probability P greater than 1/2 but less than 1?

One can say "since the legitimate receiver has the same information about the secret bit as the eavesdropper does, he cannot have any advantage over a computationally unbounded eavesdropper, so the latter will decrypt correctly with probability at least P". This is, indeed, correct. Note however that if decryption is not necessarily accurate (i.e., if decryption errors are possible), then the legitimate *sender* has an advantage over the eavesdropper since the sender, unlike the eavesdropper, knows *exactly* what the transmitted secret bit is. Therefore, if instead of making the receiver guess the transmitted bit we make the sender guess the receiver's decryption key, we may get some advantage. Thus, what we do in our scheme is:

> We make the adversary compete with the sender, not with the receiver, in contrast with the existing encryption schemes.

Competing with the sender is dramatically different from competing with the receiver because the adversary and the sender have *different goals*:

> The goal of the sender is to guess the receiver's decryption key to have him decrypt her bit correctly, whereas the goal of the adversary is to guess the sender's secret bit correctly.

Thus, the adversary and the sender may have *different probability spaces* for making their guess and therefore it is not surprising that their probabilities of success may be different. Note that *the adversary's guess of the receiver's decryption key is at least as good as that of the sender* (for information-theoretical reasons), but again – the goal of the adversary is to guess the sender's bit, not the receiver's decryption key.

We will show that it is, in fact, not too hard to arrange for the sender to have a higher probability of success (in her guessing) compared to that of the adversary, see Proposition 2 in our Section 4.2.

What is nontrivial is to have the adversary's probability of success in such a scenario to be equal to exactly 1/2, which is what we claim in our scheme.

2. INTRODUCTION

We consider a scenario where one party, Alice, wants to transmit a secret bit to another party, Bob, in the presence of a computationally unbounded (passive) adversary, Eve. We allow the legitimate parties, Alice and Bob, to fail with some controlled probability.

The way it works is roughly as follows. Bob applies a randomized (public) function F to his private decryption key b and obtains the result $B = F(b)$ that he makes public. Based on B, Alice tries to guess b. The probability to guess b is the same for Alice and Eve since they both have the same information about b in that case. However, what Eve really wants is not to recover b, but to recover Alice's bit, which means she needs to recover not the actual b, but rather what Alice thinks b is. (Think about a scenario where a customer Alice wants to transmit her credit card number to an Internet retailer Bob. Then what Eve really wants is Alice's credit card number, not Bob's decryption key.) Therefore, probability spaces for Alice and Eve are different in general, and by (privately) manipulating her probability space Alice can get advantage over Eve as far as their probabilities of success are concerned. Once again, *success for Alice (the sender) is to guess b while success for Eve is to guess the bit Alice wants to transmit to Bob.* Note that success for Alice is the same as success for Bob in the sense that Bob decrypts Alice's bit correctly if and only if Alice is successful in our terminology.

Computing exact probabilities of success for Alice and Eve theoretically can be tedious in general; we denote these probabilities by P_A and P_E, respectively. We use the following trick to make computation of P_E easy. Alice will select, with equal probability, between two mutually exclusive strategies for guessing b, thus making P_E equal to exactly $\frac{1}{2}$.

Computing P_A precisely remains a difficult theoretical task. However, in Section 4.1 we give an "existence-type" argument showing that there exists a choice of parameters that makes P_A strictly greater than $\frac{1}{2}$, see Proposition 1. Experimentally, the best we can do for P_A in a single round of the protocol is about 0.55, see our Section 8. This probability can then be amplified in different ways (see e.g. [2] or [3]); the simplest (although perhaps not the most efficient) way is described in our Section 5. It remains an interesting theoretical question though what the maximum possible value of P_A in a single round of our protocol is.

A disadvantage of our scheme is a very large expansion factor: the ciphertext for a single bit will have several thousand bits. Thus, for practical applications it makes sense to either use our scheme for public key establishment or use it in situations where only a small amount of information (e.g. a credit card number) needs to be encrypted.

3. BASIC PROTOCOL

We start with a "beta version" of our protocol to make it easier to explain its main features. The protocol is for transmitting a single secret bit from Alice to Bob.

There are (private or public) functions $f(n)$ and $g(n)$ and a public function $h(n)$ in the protocol below that have to be selected to maximize P_A, Alice's probability of guessing Bob's decryption key b. Parameters are discussed in our Section 8.

1. Bob selects, uniformly at random on integers from the interval $[0, n-1]$, a starting point b of his random walk. This b will be his private decryption key. Bob then does a simple symmetric random walk with $h(n)$ steps. Let B be the end point of Bob's random walk. If $B \geq n-1$, Bob starts over. Otherwise, Bob publishes B.
2. Step 2 is repeated by Alice m times, for a sufficiently large m.
   Alice selects, uniformly at random on integers from the interval $[B, n-1]$, a starting point a of her random walk. She then selects, with probability $\frac{1}{2}$, between $f(n)$ steps and $g(n)$ steps. Alice then does a random walk starting at the point a with the number of steps selected. Denote by A the end point of Alice's random walk. After she does her random walk, Alice moves the end point A either $\frac{1}{2}$ left or $\frac{1}{2}$ right, with probability $\frac{1}{2}$. She then moves the point a $\frac{1}{2}$ in the same direction. (This is needed to avoid situations where $a = b$ or $A = B$.)
3. Alice arranges all her m random walks at Step 2 in two groups: in one group there are walks satisfying the condition $A < B$, while in the other group there are walks satisfying the condition $B < A$. Then she selects between the two groups, with probability $\frac{1}{2}$.
4. Alice then splits all walks in the group selected at Step 3 in two groups again: in one group there are walks with $f(n)$ steps, in the other group there are walks with $g(n)$ steps. Then she selects between the two groups, with probability $\frac{1}{2}$. If the selected group turns out to be empty, Alice starts over from Step 2. If the selected group is not empty, then from this group, Alice selects one random walk uniformly at random. Let a_0 be the starting point of that selected random walk.
5. If the random walk selected by Alice at Step 4 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 $\{x < a_0\}$.
6. Alice assumes that Bob's decryption key b is in the interval she selected at Step 5 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 sends the point a_0 and the above interval labeling to Bob.
7. Bob recovers the bit corresponding to the label of the interval where his b is.

Remark 1. *At Step 2 of the above protocol Alice selects a starting point a uniformly at random on integers from the interval* $[B, n-1]$. *We note that, in fact, the distribution of a on* $[B, n-1]$ *does not have to be uniform. It can be closer to geometric, say (with points closer to B more likely to be selected). This will not affect security, but can increase the probability of correct decryption by legitimate party.*

Below we summarize public as well as private information relevant to this protocol.

Private information consists of:

Alice's choices between the options at Steps 2, 3, 4.

Alice's private key: point A (the end point of Alice's random walk).

Bob's private key: point b (the starting point of Bob's random walk).

Functions $f(n)$ and $g(n)$ can be private but they do not have to be.

Public information consists of:

Public parameters: interval $[0, n-1]$; the number $h(n)$ of steps in Bob's random walk; the number m of Alice's random walks at Step 2.

Transmitted information: point a_0 (the starting point of Alice's selected random walk).

Bob's public key: point B (the end point of Bob's random walk).

3.1. **Informal explanation.** We think it will be helpful to the reader if we give an informal explanation of what is actually going on in the above protocol. The core of the whole thing is the following non-obvious fact: if Alice and Bob do independent random walks starting at two random points, a and b, respectively, then the conditional probability $P(b < a \mid A < B < a)$ is higher when the number of steps in Alice's random walk is larger (with the number of steps in Bob's random walk fixed). Refer to our Appendix to see how to explain and theoretically quantify this statement.

Now suppose that the number $f(n)$ of steps is large while $g(n)$ is small. Then, to increase her probability of success P_A, Alice could have just done $f(n)$ steps and guess that $b < a$, conditioned on $A < B < a$. This guess would be correct with high probability. However, what Alice tries to do in our protocol is confuse Eve and make sure that Eve is unable to guess Alice's transmitted bit with probability greater than $\frac{1}{2}$. This is why Alice deliberately decreases her probability of success by selecting, in case she does $g(n)$ steps, the interval $\{x > a\}$ where the point b belongs with probability less than $\frac{1}{2}$, in the hope that her total probability of success will still be greater than $\frac{1}{2}$. This is indeed the case under appropriate choice of parameters, see our Section 8.

At the same time, the conditional probability $P(b < a \mid B < A < a \text{ or } B < a < A) = P(b < a \mid B < A \text{ and } B < a)$ "almost" does not depend on the number of steps in Alice's walk, so here Alice can have her probability of success only slightly above $\frac{1}{2}$. Nevertheless, we need to include the walks satisfying this condition in Alice's probability space to make it "symmetric" since otherwise, if we just use the walks satisfying $A < B < a$, Eve might get some idea about the number of steps in Alice's walk. Specifically, if the points B and a are far apart, then the condition $A < B < a$ makes it appear likely that the number of steps in Alice's walk was rather large than small. Symmetrizing Alice's probability space by adding walks with $B < A$ eliminates this problem, but there is a price to pay for that: the difference $P_A - \frac{1}{2}$ gets cut in half.

Finally, we note that the fact that $P(b < a \mid B < A \text{ and } B < a)$ "almost" does not depend on the number of steps in Alice's walk is in sharp contrast with the fact that $P(b < a \mid B < A)$ does strongly depend on the number of steps, see [1].

4. Probabilities of success

Recall that success for Alice (the sender) in our scenario is, given two intervals $\{x > a\}$ and $\{x < a\}$, to guess the interval where Bob's private number b is. On the other hand, success for Eve (the passive adversary) is to guess the bit Alice wants to transmit to Bob, i.e., to "guess Alice's guess" of the interval where b is. We denote by P_A and P_E the probabilities of success for Alice and Eve, respectively.

4.1. **Alice's probability P_A to guess the interval where b is.**

Proposition 1. *There exists a choice of parameters that makes P_A strictly greater than $\frac{1}{2}$.*

Proof. Recall that, while executing the protocol in Section 3 (cf. Steps 3, 4), Alice selects between two mutually exclusive options (selecting the interval $\{x < a\}$ or $\{x > a\}$) with probability $\frac{1}{2}$. Denote her probability of success if she uses the option 1 by p, and her probability of success if she uses the option 2 by q. Then $P_A = \frac{1}{2}(p+q)$. If it happens so that $p+q < 1$, then $(1-p)+(1-q) > 1$. This means that if Alice switches the interval assignments between the options, then $P_A = \frac{1}{2}((1-p)+(1-q)) > \frac{1}{2}$. This shows that there is a choice of interval assignments that gives $P_A > \frac{1}{2}$, unless $p+q = 1$ for any choice of parameters. The latter however is impossible because by varying the number of steps in a random walk for one of the two possible options, one varies the probability of guessing in this option only, see our Section 3.1 and Appendix. □

4.2. **Eve's probability P_E to guess Alice's bit.** We start by showing that $P_E < P_A$ in our scenario.

Proposition 2. *Suppose that $P_A > \frac{1}{2}$. Then $P_E < P_A$ unless Eve can determine the interval where Bob's private number b is (i.e., either $\{x < a\}$ or $\{x > a\}$) with probability exactly 1.*

Proof. Let P_{Eb} be the probability for Eve to guess the interval where b is. Denote $r = P_A,\ s = P_{Eb}$. Then Eve's probability to guess Alice's choice of interval is $rs + (1-r)(1-s) = 2rs - r - s + 1$. The difference between P_A and this probability therefore is $r - (2rs - r - s + 1) = 2r - 2rs + s - 1 = (2r-1)(1-s)$. If $r > \frac{1}{2}$ and $s \neq 1$, then this difference is positive. Thus, P_A is greater than Eve's probability to guess Alice's bit unless $s = P_{Eb} = 1$.

□

In our scenario, $P_{Eb} \neq 1$ with a reasonable choice of $h(n)$ because if, say, $h(n) > n$, then for any public point B inside the interval $[0, n-1]$, the point b can be anywhere in the interval $[0, n-1]$ with nonzero probability.

However, it is obviously preferable to have P_E not just less than P_A, but equal to $\frac{1}{2}$, the minimum possible value of P_E. This is, indeed, the case in our scheme: *after execution of the above protocol, the probability P_E for Eve to guess Alice's choice of interval is exactly $\frac{1}{2}$. This is because public information available to Eve is exactly the same for both interval assignments by Alice, and the assignments are done privately, with probability $\frac{1}{2}$.*

4.3. **If $P_E = \frac{1}{2}$, how is it possible that $P_A > \frac{1}{2}$?** Note that, given Alice's probability space $\{B < a\}$, the point B always belongs to the interval $\{x < a\}$. This implies, in particular, that if Eve selects the interval $\{x < a_0\}$ where the point B is, she will guess Alice's bit with probability $\frac{1}{2}$ because for any given point a, Alice selects between the intervals $\{x < a\}$ or $\{x > a\}$ with probability $\frac{1}{2}$.

One might ask here: why is then $P_A > \frac{1}{2}$? Since the point b is either left or right of a and Alice selects between left and right with probability $\frac{1}{2}$, then should not P_A be equal to $\frac{1}{2}$, too? An informal explanation is: the probability spaces for Eve and Alice are different. Eve only sees one point a_0, whereas Alice selects this a_0 from a pool with different points a. The point b may be left of some of these points a but right of the others, so there is no contradiction between $P_A > \frac{1}{2}$ and $P_E = \frac{1}{2}$.

The same kind of reasoning applies if Eve tries to emulate Bob: her probability space will include a *fixed* point B, which is not the case for Bob.

We note in passing that selecting the interval $\{x < a_0\}$ where the point B is will let Eve guess, with significant probability (still less than 1), the interval where the point b is. However, we remind the reader that success for Eve is not to guess b but to guess Alice's guess.

5. How to use this in real life

Given that, according to experimental results, the probability of successfully transmitting a single bit from Alice to Bob is as low as 0.55 (see our Section 8.5), a natural question now is: how can our scheme be used in real life? As we have mentioned in the Introduction, whenever there is a difference between the legitimate recipient and the adversary in the frequency of decryption errors, it is possible to transform an encryption scheme susceptible to decryption errors into one that is immune to these errors by using techniques from [2] or [3]. There is also a more simple-minded way to boost the probability of success that we describe below.

Alice can run the protocol from our Section 3 k times (every time with fresh randomness), every time transmitting the same bit c. If, say, $k = 1000$, then the probability that there will be less than 501 occurrences of c out of 1000 is $\sum_{i=0}^{501} \binom{k}{i} (0.55)^i (0.45)^{k-i} \approx 0.000846$. (This was computed using the normal approximation of the binomial distribution.)

This means that if Bob goes with the bit that has more occurrences out of k than the other bit does, he will recover Alice's bit correctly with probability at least 0.99915 if $k = 1000$. The probability of success for Bob increases with the number k of protocol runs, although increasing k obviously affects the ciphertext size.

At the same time, for Eve the expected number of times to guess Alice's bit correctly is $0.5k$, so after any k runs of the protocol there is still no reason for her to prefer one bit over the other.

6. Performance

If Alice does a random walk with, say, 100,000 steps 1,000 times, then the total number of Alice's steps will be 100,000,000. This takes about 1 minute on a regular computer (without parallelization). This is not too much but not practical either since we are talking about encrypting just a single bit.

A solution to this problem can be as follows. Alice can pre-compute a large enough private "library" of random walks, store it on her computer and access it when needed. For a single random walk, the information she needs to store includes the points a and A and the number of steps ($f(n)$ vs. $g(n)$) in that random walk. If the number of random walks in such a library is, say, 1,000,000, then storage should take about 15MB, which is not too much even for small gadgets nowadays.

Bob (the receiver) can do a similar thing: pre-compute a large enough private library of random walks, store it on his computer and select a number of walks from the library at random and make them available to Alice as needed.

With this pre-computing, the online phase of a single run of the basic protocol takes about 0.0001 sec on a regular computer, without parallelization.

7. THE REAL-LIFE PROTOCOL

Taking into account comments that we made in Sections 5 and 6, we now present a protocol, for encrypting a single bit, that can be used in real life.

As in the basic protocol of Section 3, the functions $f(n)$ and $g(n)$ can be either private or public, and the function $h(n)$ is public. The protocol description below also includes parameters n, L, R that are specified in Section 8 below. First we describe the offline phase where Alice builds a library of random walks that she will re-use to transmit multiple bits.

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 $A = B$.) This completes the offline phase.

The protocol below is for transmitting a single secret bit from Alice to Bob.

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 the 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 2 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.

7.1. **IND-CCA2 security.**

Proposition 3. *Our encryption scheme is IND-CCA2 secure.*

Proof. Recall that the ciphertext consists of several pairs (B_i, a_i) and labeling of each interval $\{x < a_i\}$ by a bit. For each particular pair (B_i, a_i), this label is Alice's secret bit with probability $\frac{1}{2}$. This is all the information available to the adversary, Eve. In particular, no matter how many (B, a) pairs with the corresponding decryption (for the same B and a) Eve collects, when she faces the same pair again she will guess Alice's secret bit with probability $\frac{1}{2}$ because we always have $B < a$ and for each particular pair (B, a) Alice selects the interval where the point B is (i.e., the interval $\{x < a\}$) with probability $\frac{1}{2}$, as we have pointed out in our Section 4.3. We note in passing that this also means that out of N pairs (B, a) (with the same B and a) accompanied by labeling of the interval $\{x < a\}$ by a bit, the expected number of labels coinciding with Alice's secret bit will be $\frac{N}{2}$. Refer to our Section 4.3 to see why this does not contradict the fact that Alice guesses the interval where Bob's point b is with probability $> \frac{1}{2}$. □

8. PARAMETERS AND COMPUTER EXPERIMENT RESULTS

Suggested parameter values for the protocol in Section 7 are: $n = 256$, $h(n) = 2000$, $g(n) = 2000$, $f(n) = 120,000$, $R = 2000$, $L = 2^{20}$.

8.1. **Size of public key.**

Bob's public key is an ordered set of R 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 of $9 \cdot 2,000 = 18,000$ bits.

8.2. **Size of private key.**

Bob's private key is an ordered set of R points b_i corresponding to the ordered set of R points B_i. Since b_i are 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.

8.3. **Ciphertext.** Alice's ciphertext (for encrypting a single bit) consists of $R = 2000$ instances of:
(a) a point a_0;
(b) labeling of the subinterval $\{x < a_0\}$ with a bit.

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

8.5. **Computer simulation results.** With $f(n) = 120,000$ steps for Alice, success rate in a single run of the loop was 76%. With $g(n) = 2000$ steps for Alice, success rate in a single run of the loop was 42%. Thus, $P_A = \frac{1}{2}(0.76+0.34) = 0.55$ for a single run. The probability of Bob successfully recovering Alice's bit after the 2,000 runs of the loop is therefore $\sum_{i=1001}^{2000} \binom{2000}{i} (0.55)^i (0.45)^{2000-i} \approx 0.999996$. (This was computed using the normal approximation of the binomial distribution.)

9. Conclusions

- We offered a public-key encryption scheme where decryption of a single bit by a legitimate party is correct with probability p that is close to 1. With suggested parameters, $p \approx 0.999996$.
- In this scheme, even a computationally unbounded (passive) adversary cannot recover the transmitted bit correctly with probability greater than 1/2. "Computationally unbounded" here includes all possible computers, quantum or not.
- The size of the receiver's public key (with suggested parameters) is on the order of 18,000 bits, whereas the size of his private (decryption) key is on the order of 16,000 bits. The size of ciphertext for encrypting a single bit is about 18,000 bits.
- The time needed to encrypt one bit is 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.

References

[1] M. Bessonov, D. Grigoriev, V. Shpilrain, *Probabilistic solution of Yao's millionaires' problem*, preprint. https://eprint.iacr.org/2017/1129

[2] C. Dwork, M. Naor, O. Reingold, *Immunizing Encryption Schemes from Decryption Errors*, in: Advances in Cryptology – EUROCRYPT 2004, Lecture Notes Comp. Sc. **3027** (2004), 342–360.

[3] T. Holenstein and R. Renner, *One-Way Secret-Key Agreement and Applications to Circuit Polarization and Immunization of Public-Key Encryption*, in: Advances in Cryptology – CRYPTO 2005, Lecture Notes Comp. Sc. **3621** (2005), 478–493.

Appendix

9.1. **How $P(b < a | A < B < a)$ depends on the number of steps.** Let $\alpha > 0$ and n^α be the number of steps in Alice's walk and suppose initially that this number is odd (to avoid parity issues, although the conclusion that $P(b < a | A < B < a)$ depends on α still holds when the number of steps is even). Let n^β be the fixed number of steps in Bob's walk with $0 < \beta < 2$. Then $P(b < a | A < B < a)$ depends on α as follows:

- When α is very small, $P(b < a|A < B < a)$ is very close to $1/2$.
- As α increases, $P(b < a|A < B < a)$ tends to $P(b < a|B < a)$, which tends to 1 as $n \to \infty$.

Suppose that $n^\alpha = 1$. Then $P(b < a|A < B < a)$ is the probability that $b < a$, given that Alice's one step was to the left and Bob's final location happens to be between A and a, for which there is only one possibility $B = a - 0.5$ and $A = a - 1$. In this case,

$$P(b < a|A < B < a) = P(b < B) = \frac{1}{2} - O\left(n^{-\beta/2}\right),$$

or, if we remove the possibility that $B = b$, by shifting Bob's end point by adding or subtracting 0.5 with equal probability, then

$$P(b < a|A < B < a) = P(b < B) \xrightarrow{n\to\infty} \frac{1}{2}$$

The probability is not exactly equal to $1/2$ due to the restriction that $0 \le b \le n-1$ and $B < n-1$. However, as $n \to \infty$, the probability that b is close to 1 or n goes to zero. As α increases, given that $B < a$, B is more likely to be farther from a, and when B is farther from and to the left of a, b is more likely to be less than a. This is because the number of steps in Bob's walk remains fixed, and Bob is (almost) equally likely have started to be to the left or to the right of B. If $b < B$, certainly $b < a$. If $B > b$, the fact that $A - a$ can be larger, increases the probability that $B < b < a$. "Almost" because of the restriction on b and B mentioned above.

Now, as α increases, the condition $A < B < a$ implies that A will be farther from a. Eventually, for α large enough, A will be outside of the interval $\{0, 1, \ldots, n-1\}$ with probability close to 1. The probability of A being in the interval will be exponentially small in α. If A is outside of this interval, then $P(b < a|A < B < a) = P(b < a|B < a$, Alice's walk ends to the left of her starting point$) = P(b < a|B < a)$.

Lemma 1. *$P(b < a|B < a) \to 1$ as $n \to \infty$.*

To see that this is true, consider

$$P(b < a|B < a) = P(b < B < a|B < a) + P(B < b < a|B < a)$$

The first term, $P(b < B < a|B < a) = P(b < B) \to 1/2$ as $n \to \infty$. If B is distance $O(n^{\beta/2+\varepsilon})$ for small $\varepsilon > 0$, $P(b < B)$ goes to $1/2$ as $n \to \infty$, as it is just the probability that the endpoint of the walk is to the right of the starting point. If B is close to 0, the probability is under $1/2$ since Bob's starting point b is restricted to $\{0, 1, \ldots, n-1\}$. As $n \to \infty$, the probability that B is close to 0 goes to zero. If B is close to $n-1$, $P(b < B)$ is actually close to 1, but the probability that B is close to $n-1$ also goes to zero.

The second term, $P(B < b < a|B < a) \xrightarrow{n\to\infty} 1/2$ as well. Here, we consider two possibilities:

- $P(B < b < a|B < a, a - B \ge n^{\beta/2+\varepsilon}) \xrightarrow{n\to\infty} 1/2$, since the probability of the displacement being greater than $O(n^{\beta/2})$ is exponentially small.
- $P(B < b < a|B < a, a - B < n^{\beta/2+\varepsilon})$ is not close to 1, however,

$$P\left(a - B < n^{\beta/2+\varepsilon}\right) \xrightarrow{n\to\infty} 0$$

From this,

$$P(B < b < a|B < a) \xrightarrow{n\to\infty} P(B < b < a|B < a, a - B \ge n^{\beta/2+\varepsilon}) \xrightarrow{n\to\infty} 1/2.$$

9.2. **Why $P(b < a | B < A < a$ or $B < a < A)$ does not depend greatly on the number of steps.** Consider the two events in the condition separately and note that they are disjoint.

- If $B < a < A$, then the probability that $b < a$ does not depend on Alice's walk, and thus on α, at all, since the condition is that Alice ended to the right of her starting point a (the probability of which is the same as Alice ending to the left of a) and B is always to the left of a in our setup. Note also that in the sample space consisting of the events $\{B < a < A\} \cup \{B < A < a\}$, the event $\{B < a < A\}$ has probability greater than $1/2$ since Alice is more likely to end to the right of A with no other restriction than to the left of a but to the right of B.
- If $B < A < a$, the probability that $b < a$ does depend on the number of steps in both walks, however, if $\beta < \alpha$ are fixed, the probability will approach 1 as $n \to \infty$. Thus, the dependence on α is weak, so long as $\beta < \alpha$. We have here that under this condition, Alice ended her walk to the left of where she started, and Bob ended to the left of Alice's endpoint. If Alice performed a greater number of steps than Bob, to not have $b < a$, Bob's displacement would have to be greater than Alice's.