Secure information transmission based on physical principles

Dima Grigoriev[1] and Vladimir Shpilrain[2]

[1] CNRS, Mathématiques, Université de Lille, 59655, Villeneuve d'Ascq, France
dmitry.grigoryev@math.univ-lille1.fr
[2] Department of Mathematics, The City College of New York, New York, NY 10031
shpil@groups.sci.ccny.cuny.edu *

Abstract. We employ physical properties of the real world to design a protocol for secure information transmission where one of the parties is able to transmit secret information to another party over an insecure channel, without any prior secret arrangements between the parties. The distinctive feature of this protocol, compared to all known public-key cryptographic protocols, is that neither party uses a one-way function. In particular, our protocol is secure against (passive) computationally unbounded adversary.

1 Introduction

In public-key cryptography, one of the basic functionalities is *encryption*, which is performed over an open channel, without any prior secret arrangements between the parties engaged in a cryptographic protocol.

More specifically, the scenario is as follows. One of the parties, Alice, wants to transmit to another party, Bob, some secret information (say, a secret number m). She can use an encryption algorithm to produce an encryption $E(m)$ of her secret number m. She then transmits $E(m)$ to Bob over an open (i.e., insecure) channel, and Bob uses a decryption algorithm to recover m from $E(m)$. All algorithms and all transmissions are supposed to be known to the public and yet the cryptosystem should be secure; this is the so-called *Kerckhoffs's principle*, which is considered mandatory in public-key cryptography. Therefore, in particular, Bob's decryption algorithm should involve a private key that only Bob knows. Also, the encryption function $m \to E(m)$ should be *one-way*, which means, informally, that it is efficient to compute but hard to invert (at least, on most inputs). "Efficient" here means that the function of an input can be computed in time polynomial in the size of an input. Thus, "hard to invert" roughly means that there is no algorithm of time complexity polynomial in the size of an input that would find a pre-image of any given input.

It was shown in [4] that for a public-key encryption protocol to be secure, it has to employ at least one one-way function. On the other hand, a necessary

* Research of the second author was partially supported by the NSF grants DMS-0914778 and CNS-1117675

condition for the existence of one-way functions is that **P** is not equal to **NP**, where **P** is the class of functions computable in deterministic polynomial time, and **NP** is the class of functions computable in non-deterministic polynomial time. It is a central open problem in complexity theory whether or not these two classes are equal, with the prevalent opinion being that they are not. See [2] for a general background on this famous problem.

One immediately obvious thing is that a definition of any complexity class depends on how one formalizes the concept of being "computable". The universally accepted formalization now is that a function is computable if there is a *Turing machine* that computes this function. Again, see [2] for a general background on Turing machines; here we just say that a Turing machine is a simple mechanism that manipulates symbols on a strip of tape according to a table of rules. This machine was described in 1936 by Alan Turing. A Turing machine that is able to simulate any other Turing machine is called a *universal Turing machine*. A more mathematically oriented definition with a similar "universal" nature was introduced by Alonzo Church, whose work on lambda calculus intertwined with Turing's in a formal theory of computation known as the Church-Turing thesis. The thesis states that Turing machines are indeed a good formalization of the intuitive notion of computability and provide a precise definition of an algorithm.

It is the main purpose of this paper to show that there are secure encryption protocols that do not employ any one-way functions. On the intuitive level, when combined with the aforementioned results, this yields a feeling that there are "naturally computable" functions that cannot be computed by a Turing machine, but we do not make any formal statements in that direction at this time.

We note that our procedures are rather practical and only use physical properties of objects encountered by people in everyday life.

The structure of the paper is as follows. In Section 2, as a "warm up", we show how to use a natural algorithm for a special case of a cryptographic primitive called *secret sharing*. More specifically, our special case is: a dealer distributes "shares" of a secret (which could be a secret number) to two parties, the idea being that only when the two parties get together, they can recover the whole secret, but neither alone can. This problem may seem trivial: if the secret number is n, then the dealer can just split n as $n = n_1 + n_2$ and give n_1 to one of the parties and n_2 to the other. What we want to achieve however is that, after the two parties have recovered the whole secret, they end up still not knowing the other party's share. This is obviously impossible if the secret is $n = n_1 + n_2$ and the shares are n_1 and n_2. In our Section 2, we show how to solve this problem by using a natural algorithm that does not, in fact, employ any non-trivial physics; just everyday experience. It is unknown (at least, to us) whether this problem can be solved by a procedure implementable on a Turing machine. What is interesting though is that this question is equivalent to the following one, that looks almost purely mathematical. Alice knows a point (x_1, y_1) in the plane, while Bob knows a point (x_2, y_2). Can they exchange information in such a way

that each ends up knowing the slope $s = \frac{y_2-y_1}{x_2-x_1}$ of the line connecting their points, but neither ends up knowing the other party's point (x_i, y_i) ? We note that it is fairly easy to arrange secret sharing between more than two parties without any party having to reveal their share to recover the secret, see [3] or [5]. At the end of Section 2, we give an application (pointed out to us by Bren Cavallo) of our method to Yao's "millionaires' problem" [7].

Then, in Section 3, we describe an encryption protocol where a secret number is encrypted as the length of a piece of an elastic rope, which consists of two parts with different modula of elasticity, one of them being secret. Again, physical properties that are used in this protocol are very simple, and the protocol can be easily implemented in real life, although it is only suitable for transmitting secret information over a short distance. What is important is that this protocol is secure even against (passive) computationally unbounded adversary. Since a computationally unbounded adversary can use "encryption emulation attack" (i.e., reproduce encryption algorithm with all possible randomness and then compare the results to the actual encrypted message), security against such an adversary can only mean one thing: there are many possible combinations of different messages with different random choices that result in the same encryption. This is exactly what happens in our protocol.

We admit that the encryption protocol in Section 3 is rather impractical, but we believe it is helpful in understanding why what we call "nature-based cryptography" provides for what the "usual", complexity-based, cryptography cannot possibly provide: security against (passive) computationally unbounded adversary.

In Section 4, we give a more detailed analysis of the encryption emulation attack in our scenario, and in Section 5, we discuss security against *active* adversary, i.e., an adversary who does not just observe and process information, but can interfere with the protocol itself.

Finally, in our main Section 6, building on the ideas of Section 3, we give a rather simple and practical encryption protocol where instead of a piece of rope we use an electrical circuit to transmit data.

2 Secret sharing between two parties

The problem that we address in this section is as follows. The dealer wants to distribute information ("shares") about a secret number s to two parties, Alice and Bob, in such a way that only when the two parties get together, they can recover the whole secret, but neither alone can.

One possible solution (which is a special case of Shamir's solution [6] of a more general problem) is: the dealer represents the number s as $s = \frac{y_2-y_1}{x_2-x_1}$ and gives (x_1, y_1) to Alice and (x_2, y_2) to Bob. Problem solved.

Now we want to have an additional functionality: after the two parties have recovered the whole secret s, they end up still not knowing the other party's share. It is an open problem whether or not this can be achieved by "purely

mathematical" methods (implementable on a Turing machine), but what we offer here is a solution that uses physical properties of the real world.

We therefore position Alice and Bob in a plane, Alice at the point $A = (x_1, y_1)$ and Bob at the point $B = (x_2, y_2)$, and give them both long pieces of rope. We assume that the scaling is such that Alice cannot see Bob's point from where she is and Bob cannot see Alice's point from where he is. Now here is the protocol:

1. Alice fixes one end of her rope at the point $A = (x_1, y_1)$ and selects a neighborhood U of the point A that cannot be seen by Bob. Bob, too, selects a private neighborhood V of his point $B = (x_2, y_2)$.
2. Alice throws (or brings) the other end of her rope to a random point C in the plane, far enough so that her neighborhood U could not be seen from C. Then she positions the part of the rope inside U so that this part is not a straight line. She then communicates the coordinates of the point C to Bob.
3. Bob walks to the point C, ties one end of his rope to Alice's rope, then walks back to his point B, unwinding his rope along the way. That is, he is not pulling the rope at this step, just unwinding.
4. When Bob reaches his point B, he starts pulling the rope until Alice tells him to stop, which is as soon as Alice sees that the part of the rope inside her neighborhood U is a straight line.
5. To make sure that it is not by accident that the part of the rope inside her neighborhood U is a straight line, Alice asks Bob whether or not the part of the rope inside his neighborhood V is a straight line. If it is not, then Alice starts pulling her end of the rope toward her point A until Bob tells her to stop, which is as soon as Bob sees that the part of the rope inside his neighborhood V is a straight line.
6. When the parts of the rope inside both neighborhoods U and V are straight, Alice and Bob assume that their points A and B are connected by a straight rope, and they find the slope s of the corresponding straight line by selecting any two points on the parts of the line inside their private neighborhoods.

Some parts of this protocol may seem redundant, and indeed, if both parties are honest, the protocol can be simplified. However, we want to make sure that if one of the parties is dishonest, he/she cannot cheat to get a hold of the other party's share (x_i, y_i). This is why, in particular, Alice tells Bob to stop as soon as she sees that the part of the rope inside her neighborhood U is a straight line. Otherwise, Bob could triangulate Alice's point A by straightening the rope between A and two different points of his choice.

We note that even though neither party in this scenario can determine the other party's point precisely, the slope of the line connecting the two points gives away *some* information, and this has the following interesting application to Yao's "millionaires' problem".

2.1 Application to Yao's "millionaires' problem"

The "two millionaires problem" introduced in [7] is: Alice has a number x_1 and Bob has a number x_2, and the goal of the two parties is to solve the inequality $x_1 < x_2$? without revealing the actual values of x_1 or x_2.

Here is how we can solve this problem by using the solution of the secret sharing problem above. Alice privately selects a number $y_1 < 0$, and Bob privately selects a number $y_2 > 0$. Now Alice has a point (x_1, y_1) in the plane, and Bob has a point (x_2, y_2) in the plane. By using the method in this section, Alice and Bob can determine the slope s of the line connecting the two points without revealing the actual points. Then $x_1 < x_2$ if and only if $s > 0$.

Note that Yao's "millionaires' problem" in this interpretation is a formally weaker problem than secret sharing between two parties because to solve the "millionaires' problem", we do not need to know the actual value of the slope s; it is sufficient to know just the sign of s.

3 Encryption without one-way functions

Disclaimer. *We realize that the encryption protocol in this section is rather impractical, but we believe it is helpful in understanding why what we call "nature-based cryptography" provides for what the "usual", complexity-based, cryptography cannot possibly provide: security against (passive) computationally unbounded adversary. In particular, we believe that "pulling the rope" relays a helpful visual imagery of how the receiver (Bob) participates in the transmission: he is not just "sitting there" waiting for information from Alice to arrive, but participates in transmission actively by "pulling out" a secret from Alice's private space. This is something that seems to be impossible to mimic in the "usual" public-key scenario, and it is there where physical properties of the real world play a crucial role.*

For a simple and practical implementation of the ideas of this section, we address the reader to our Section 6, but we do recommend to read this and the following two sections first.

In this section, we give an encryption protocol that does not use any one-way functions, so that its security is based on physical properties of real-world objects. Particular real-world objects that we use in this section are elastic ropes that quickly regain their original ("natural") length after being stretched by a force and then released. We also assume that all ropes mentioned in this section obey Hooke's law when deformed:

$$\Delta l = \frac{F}{E} \cdot l,$$

where l is the natural length of a rope, Δl is the rope's extension (strain), E is the rope's modulus of elasticity, and F is the *normal stress*, i.e., the force straining the rope divided by the area of the rope's cross-section. We note that in physics, a more popular notation for the normal stress is σ, but we believe that in our situation, F is a more reader-friendly notation, and we shall often refer to F simply as "force" because the thickness of the rope does not play any role in our considerations.

The scenario is as follows. Alice wants to send a secret positive number x_A to Bob. We assume that Alice has a private space U (e.g. a private room) where

nobody (i.e., neither Bob nor an eavesdropper Eve) can observe her actions. Similarly, Bob has a private space V where nobody can observe his actions. Alice and Bob share a long elastic rope: Alice holds one end of the rope, and Bob holds the other. We assume that Alice also has a collection of ropes of various modula of elasticity *known to everybody* (this is a tribute to Kerckhoffs's principle) that she can combine privately (by cut and paste) to produce two pieces of rope, R and R', of the same private random length but with different private random modula of elasticity.

We assume that everybody (Alice, Bob, Eve) can observe (and measure) everything that is going on in the "public space", i.e., outside the union of U and V. For simplicity, we are assuming that both U and V are convex domains in the plane.

Now here is the protocol itself:

1. Alice begins by cutting off a piece of the rope of random length inside her private space U. Bob does the same inside his private space V. Now nobody knows the total natural length of the rope connecting Alice and Bob.
2. Alice replaces, inside her private space U, a random part of the rope by her private piece of rope R with private random modulus of elasticity. The point of doing this is that now the eavesdropper Eve does not know the modulus of elasticity of the whole rope connecting Alice and Bob.
3. Alice interprets her secret number x_A as the length of the part of the rope which is inside her private space U in the beginning of the transmission. (We are assuming that a straight rope of length x_A can fit inside U). Specifically, she randomly splits x_A as a sum of two positive numbers: $x_A = x'_A + \varepsilon$, and pins the rope to the floor so that the part of the rope between the rope's end and the pin has length ε, and the part of the rope between the pin and the boundary of U has length x'_A. She also marks the point P_A on the rope where the rope intercepts the boundary of U.
   Denote by x_B the natural length of the part of the rope inside Bob's private space V, and by L the natural length of the part of the rope outside the union of U and V. We assume that everybody (Alice, Bob, Eve) knows L.
4. Now the transmission begins. Bob, who remains inside his private space V, pulls the rope on his end with a private force F_B (he may randomly change its magnitude along the way), until the marked point P_A is at a random point inside his private space V. (This prevents the eavesdropper Eve from knowing the natural length of the part of the rope that Bob has pulled inside V since the beginning of the transmission.) It is also important that during this whole procedure, no part of Alice's private piece of rope R gets outside of U, to prevent Eve from computing its modulus of elasticity.
5. Alice replaces the piece of rope R by R', to prevent Eve from computing the modulus of elasticity of the rope used for transmission. She then releases her pin and tells Bob to pull the entire rope inside his private space V.
6. Having done that, Bob measures the natural length of the entire rope, subtracts L and x_B, and gets x_A, which is Alice's secret number.

Security of this protocol is essentially based on the following claim:

Main claim. If the eavesdropper Eve does not know the modulus of elasticity of the rope inside Alice's private space U, then she cannot unambiguously determine the natural length of any part of the rope pulled from the inside of U to the outside.

Proof. The validity of this claim follows from Hooke's law: $\Delta l = \frac{F}{E} \cdot l$, where l is the natural length of a rope, F is the force straining the rope, Δl is the rope's extension (strain), and E is the rope's modulus of elasticity.

What Eve wants to know here is l, the natural length of the rope (or part of it) inside U. She is observing Δl, but E is private, so even if Eve is able to measure the force F, she still has two unknowns in this equation, which we can re-write as: $\frac{F}{\Delta l} = \frac{E}{l}$. If Eve knows F and Δl, then she knows the ratio of E and l, which still leaves many possibilities for different pairs (E, l) with the same ratio. Therefore, Eve cannot determine l unambiguously by observing the "transmission". □

In Section 5, we discuss the scenario where the adversary can actively interfere with the protocol.

4 Encryption emulation attack

Encryption emulation attack is reproducing encryption algorithm with all possible randomness and then comparing the results to the actual encrypted message. In the classical model, encryption is a (possibly non-deterministic) one-to-one function (otherwise, decryption by the receiver would not be unique), and therefore encryption emulation attack is always effective. This is usually the most powerful attack, but the problem is: this attack typically requires prohibitively vast amount of resources from the attacker. If, however, the attacker is computationally unbounded, she can use this attack, hence no encryption scheme in the classical model can be secure against computationally unbounded adversary.

Since in our scheme, we claim security against computationally unbounded adversary, this can only mean one thing: our encryption is not one-to-one, in contrast with the classical model. This yields a question of how can decryption by the receiver in this case be unique. Let us compare our model to the classical one in more detail.

The main difference is that in our model, the receiver (Bob) actively participates in Alice's transmission, in contrast with the classical model where the receiver is a passive observer of the transmission, just like Eve, and therefore Eve and Bob get exactly the same information during the transmission phase of the protocol. In our scheme, on the other hand, Bob gets more information during the transmission phase, namely he knows where the point P_A ends up, while Eve does not. In other words, Bob influences the transmission phase with his secret key. It would be interesting to arrange an encryption scheme with similar properties by purely mathematical means, but at the time of this writing we do not know whether or not this is possible (most likely, it is not).

Now let us see why the encryption in our scheme is not one-to-one as far as Eve is concerned. The point is, again, that encryption is a function not only of Alice's secret number x_A and her other secret parameters ε, E_A (where E_A is her secret modulus of elasticity), but also of Bob's secret parameters that include his force F_B and the final position $B(P_A)$ of the point P_A. To simplify the notation, denote by X the set of Alice's parameters, and by Y the set of Bob's parameters. Now the encryption of Alice's secret number x_A is a function $F(X, Y)$. First of all, we have to ask ourselves: what is the co-domain of this function, i.e., where does it take its values? One possible answer to this question is that a value of F is a function $S(t)$ that describes the state of the rope at a moment t after the beginning of the protocol, where t changes from 0 to the moment when the whole rope is in Bob's private space V. Denote by $\hat{S}(t)$ the "retract" of this function, i.e., the state of the rope observable by Eve at a moment t.

Since we claim security against computationally unbounded adversary, we do not care whether the function F is one-way or not (in fact, it is not); what we care about is that this function is not one-to-one as far as Eve is concerned, in the sense that if, for some $\hat{S}(t)$, there is a pair (X_1, Y_1) such that $F(X_1, Y_1) = \hat{S}(t)$, then there is also at least one pair (X_2, Y_2) such that the corresponding secret numbers $x_A^{(1)} \in X_1$ and $x_A^{(2)} \in X_2$ are different, and yet

$$F(X_1, Y_1) = F(X_2, Y_2).$$

At the same time, the function F should be one-to-one as far as decryption by Bob is concerned, i.e., for any $S(t)$, (X_1, Y), (X_2, Y) such that $F(X_1, Y) = F(X_2, Y)$, one should have the corresponding secret numbers x_A equal. This latter condition is satisfied by the very design of the protocol since we have seen in Section 3 that Bob decrypts without any ambiguity.

To show the validity of the displayed condition, we note that if we change ε (which is part of x_A), the function $S(t)$ will not change at least until the moment when Alice releases her pin. For $\hat{S}(t)$ to remain unchanged after Alice releases her pin, the total natural length of the rope visible to Eve after the pin is released should not change. That is, if, for example, we decrease ε by some δ, there should be a way to increase the length of the rest of the rope visible to Eve by the same δ, by varying other parameters. Indeed, recall Hooke's law again: $E \cdot \Delta l = F \cdot l$, where l is the natural length of the part of the rope visible to Eve after the pin is released, without the part of length ε (because the latter part is not being strained). Thus, the total natural length of the rope visible to Eve after the pin is released is $l + \varepsilon$.

Now suppose $\varepsilon' = \varepsilon - \delta$. Then, to keep the total natural length visible to Eve unchanged, the length l has to be replaced by $l' = l + \delta$. Then, in order to keep Δl (visible to Eve during the transmission) unchanged, Alice's private elasticity modulus E can be changed to E' so that $E' \cdot \Delta l = F \cdot l'$. We can solve this for E': $E' = F \cdot \frac{l'}{\Delta l}$, thereby showing that different pairs (E, x_A) can produce the same state of the rope observable by Eve. This is why Eve cannot decrypt unambiguously by using the encryption emulation attack.

5 Active adversary

Active adversary is an adversary who can actively interfere with the protocol, as opposed to just observing and analyzing. We distinguish two kinds of active adversaries:

1. (not-so-dangerous kind) An adversary of this kind would just mess up communication between Alice and Bob without attempting to retrieve Alice's secret message. The simplest way would be just to cut the rope. There is no defense against such malicious behavior (after all, an Internet cable can be cut, too), but there is no threat to security either, so we shall not be concerned about adversaries of this kind.

2. (dangerous kind) An adversary of this kind would attempt to retrieve Alice's secret message. In our situation, an adversary may try to measure the natural length of a piece of the rope emerging from Alice's private room by applying a force compensating Bob's force to return a selected piece of the rope to its natural state. In this case, Bob will detect a counter-force and abort the protocol.

Thus, even though Alice and Bob may not be able to prevent interference in their protocol, they are able to detect malicious activity and abort the protocol to prevent an adversary from recovering secret information. This has a superficial resemblance to the intruder detection ideas in quantum cryptography [1], although our scenario is much closer to everyday life. Also, in the scenario in [1], the intruder may not be detected (with nonzero probability) if she makes correct guesses.

6 A more practical implementation: electrical circuit

In this section, we describe a more practical implementation of the ideas of Section 3. Obviously, using a piece of rope for transmission of data limits applicability of relevant encryption essentially to communication between two offices on the same floor (at best). Therefore, even though the ideas look nice in theory, if they are to be applied to real-life communication, their implementation has to be much more practical.

Here we suggest an encryption protocol similar to that of Section 3, but instead of a piece of rope we use an electrical circuit to transmit data, and the secret information that we transmit is the electric charge of a capacitor.

Thus, the initial setup is as follows. Alice wants to send a secret positive number q_A to Bob. We assume that Alice has a private space U (e.g. a private room) where nobody (i.e., neither Bob nor an eavesdropper Eve) can observe her actions. Similarly, Bob has a private space V where nobody can observe his actions. In her private space U, Alice has a capacitor C_1 of the capacitance c_A with the charge q_A, while Bob in his private space V has a capacitor C_2 of the capacitance c_B with the charge q_B, selected by Bob randomly. These capacitors are connected to form an electrical circuit (see Figure 1), in such a way that the capacitors' plates holding positive charge are connected by one wire, and

the plates holding negative charge are connected by another wire. Alice also has a switch that keeps the circuit disconnected until the actual transmission begins, and she has an ammeter to monitor the electric current in the circuit. Bob also has (in his private space) a *rheostat* (i.e., a variable resistor) included in the circuit. This allows him to randomly change the resistance of the whole circuit, and therefore also to change parameters of the electric current during the transmission. Now more formally:

Alice's (sender's) public key: capacitance c_A

Alice's secret message: charge q_A

Bob's (receiver's) long-term private key: capacitance c_B

Bob's session private key: charge q_B. This private key is selected by Bob randomly before each transmission from Alice.

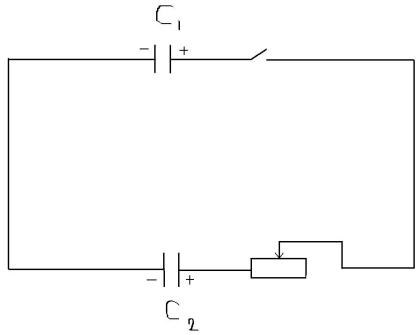

Fig. 1. Electrical circuit

Now here is the protocol itself:

1. Alice uses her switch to connect the circuit, thereby starting re-distribution of electric charges between the capacitors C_1 and C_2. When the re-distribution of the charges is complete, Alice's ammeter shows that there is no current in the circuit, so she disconnects the circuit.
2. After re-distribution of the charges is complete, let Q_A be the new charge of the capacitor C_1, and Q_B the new charge of the capacitor C_2. Then:

$$Q_A = c_A \cdot \frac{q_A + q_B}{c_A + c_B}, \qquad Q_B = c_B \cdot \frac{q_A + q_B}{c_A + c_B}, \qquad \frac{Q_A}{c_A} = \frac{Q_B}{c_B}.$$

3. Bob, who knows Q_B, c_B, q_B, and c_A, can now recover Alice's secret q_A from the second formula above: $q_A = Q_B \cdot (1 + \frac{c_A}{c_B}) - q_B$.

Encryption emulation attack on this protocol will not work for the same reason it does not work with the protocol in our Section 3. Namely, in the three equations displayed at Step 2 of our protocol above, there are 5 parameters unknown to the adversary (the only known parameter is c_A), which yields many

possible solutions for q_A. In fact, one of the three equations is redundant (for example, equation 1 follows from the equations 2 and 3), so there are just 2 equations with 5 unknowns to the adversary!

The adversary may attempt to measure the electric current I in the circuit during the transmission and use other laws of physics to try to recover some of the parameters. Relevant laws of physics include Ohm's law $I = \frac{U}{R}$, where U is the voltage and R the total resistance in the circuit. Since right after Alice turns her switch on, the initial voltage U is $\frac{q_A}{c_A} - \frac{q_B}{c_B}$, upon combining these two formulas we get $I = \frac{q_A}{c_A R} - \frac{q_B}{c_B R}$.

This formula introduces two more parameters, at least one of which, the total resistance R, cannot be measured by the adversary Eve because some parts of the circuit, including the rheostat with variable resistance, are hidden from Eve.

There are other formulas involving some of the parameters of our electrical circuit, but they all are similar to the formula above in the sense that they introduce new parameters, at least one of which cannot be evaluated by Eve because it is relevant to properties of those circuit elements that are hidden in either Alice's or Bob's private space.

For the same reason, even an active adversary cannot determine q_A in this situation because even if she measures, say, I, at whatever moment(s) during the transmission, it will not help her because she does not know how the rheostat's resistance (in Bob's private space) changes. Of course, an active adversary can mess up things in many different ways (the simplest way being just cutting a wire); in particular, she can make sure that Bob will not get the right value of q_A in the end by connecting her own capacitor to the circuit. Detecting this or other kind of interference with the protocol is one of the issues to be addressed in a real-life implementation of our protocol, but we leave this discussion out of the present paper since it would take us too far into the realm of electrical engineering. What we emphasize here is that even an active adversary cannot get a hold of the actual secret q_A.

6.1 Attempting to compromise the receiver's long-term private key

Suppose Alice transmits several secret messages to Bob. Then, in addition to the three equations (see Step 2 of our protocol), the adversary gets several other triples of equations, with the same c_A and c_B, but different other parameters. (As we have seen before, one equation in each triple is redundant, so the adversary actually gets extra *pairs* of equations.) Thus, with each new pair of equations the adversary gets 4 new unknowns (q_A, q_B, Q_A, Q_B), which does not help her recover any of the unknowns, including Bob's long-term private key c_B.

Another attack, called *chosen ciphertext attack*, which is typically considered in cryptography, allows the adversary to encrypt messages of her choice, in an attempt to get information about a long-term private key. What it means for our protocol is that in the system of equations at Step 2 of our protocol there would be 4 unknowns to the adversary rather than 5, but this still is not enough to determine any of the unknowns without ambiguity. Also, if the adversary

encrypts messages of her choice more than once, then each time she will get 2 new equations with 3 new unknowns, which does not help her.

Finally, suppose Alice herself attempts to get Bob's long-term private key. In this case, she knows q_A, Q_A, c_A, but does not know q_B, Q_B, c_B, and this still does not allow her to recover Bob's long-term private key c_B without ambiguity. Encrypting several messages in this scenario will produce more pairs of equations together with new pairs of unknowns q_B, Q_B, which still does not help her.

7 Conclusions

We have employed physical properties of the real world to design public-key encryption protocols secure against computationally unbounded adversary, which is impossible in the "usual" (i.e. complexity-based) cryptography.

What appears to be the main reason why this is possible in nature-based cryptography but impossible in complexity-based cryptography is that physical properties of the real world can be employed to arrange for the receiver (Bob) to be able to influence the transmission of information from the sender (Alice) by using his *private key*, as opposed to the typical scenario in complexity-based cryptography where Bob, after having published his *public key*, is just "sitting there" waiting for information from Alice to arrive.

Acknowledgements. Both authors are grateful to the Max Planck Institut für Mathematik, Bonn for its hospitality during their work on this paper. We are also grateful to Igor Monastyrsky for consultations on physical aspects of our schemes, and to Bren Cavallo for pointing out an application of our method to Yao's "millionaires' problem". The first author is also grateful to Labex CEMPI (ANR-11-LABX-0007-01).

References

1. C. H. Bennett and G. Brassard, *Quantum Cryptography: Public key distribution and coin tossing*, in Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, Bangalore, p. 175 (1984).
2. M. Garey, J. Johnson, *Computers and Intractability, A Guide to NP-Completeness*, W. H. Freeman, 1979.
3. D. Grigoriev and V. Shpilrain, *Secrecy without one-way functions*, Groups, Complexity, Cryptology **5** (2013), 31–52.
4. R. Impagliazzo and M. Luby, *One-way functions are essential for information based cryptography*, in Proc. 30th IEEE Sympos. on Found. of Comput. Sci., IEEE, New York, 1989, pp. 230-235.
5. D. Kahrobaei, M. Habeeb, V. Shpilrain, *A secret sharing scheme based on group presentations and the word problem*, Contemp. Math., Amer. Math. Soc. **582** (2012), 143–150.
6. A. Shamir, *How to share a secret*, Comm. ACM **22** (1979), 612-613.
7. A. C. Yao, *Protocols for secure computations* (Extended Abstract), 23rd annual symposium on foundations of computer science (Chicago, Ill., 1982), 160–164, IEEE, New York, 1982.