PUBLIC KEY ENCRYPTION AND ENCRYPTION EMULATION ATTACKS

DENIS OSIN AND VLADIMIR SHPILRAIN

Abstract. The main purpose of this paper is to show that public key encryption can be secure against the "encryption emulation" attack (on the sender's encryption) by computationally unbounded adversary, with one reservation: a legitimate receiver decrypts correctly with probability that can be made arbitrarily close to 1, but not equal to 1.

1. Summary of our claims

We thought it would make sense to summarize, for the reader's convenience, our two main claims in a separate section, before proceeding to a narrative introduction.

In Section 3, we describe a public-key encryption protocol that allows Bob (the sender) to send secret information to Alice (the receiver), encrypted one bit at a time, so that:

(1) Assuming that Eve (the adversary):
(a) is computationally unbounded,
(b) knows everything about Bob's encryption algorithm and hardware,
(c) does not know Alice's algorithm for creating public key, then:
she cannot decrypt any single bit correctly with probability $> \frac{3}{4}$ by emulating Bob's encryption algorithm.

We note that the assumption (c) is not in line with what is called "Kerckhoffs' assumptions", or "Kerckhoffs' principle" (see e.g. [3]), which is considered mandatory in practical cryptography. However, the assumption (a) is not encountered in real life either, so this claim of ours should be considered from a purely theoretical point of view, although we speculate in the end of the Introduction that this claim may be of interest in real-life non-commercial cryptography.

The second claim *is* in line with Kerckhoffs' assumptions.

(2) Assuming that Eve:
(a) knows everything about Alice's and Bob's algorithms and hardware (Kerckhoffs' assumptions),
(b) is exponentially, but not superexponentially, computationally superior to Alice and Bob, then:
she cannot decrypt any single bit correctly with probability significantly higher than $\frac{3}{4}$ by using the encryption emulation attack in the broad sense (i.e., where she can emulate both the sender's and the receiver's algorithms).

We do not want to be too formal here about what "exponentially, but not superexponentially, computationally superior" means. Intuitively, the reader can think of the following interpretation: if Alice and Bob are capable of performing at most n operations per second, then Eve can perform at most C^n operations per second for some fixed constant C independent of n.

Research of the first author was partially supported by the NSF grant DMS-0605093. Research of the second author was partially supported by the NSF grant DMS-0405105.

More specifically, we show that there is a k-step algorithm for the receiver (Alice) to obtain her public key that takes time $O(k^3)$, whereas the adversary who would like to emulate this algorithm with all possible randomness would have to take time $O(k^k)$.

Our focus in this paper is on claim (1) because, in our opinion, it is more interesting from the theoretical point of view.

2. Introduction

In this paper we continue to explore how non-recursiveness of a decision problem (as opposed to computational hardness of a search problem) can be used in public key cryptography. This line of research was started in [6] (note that the earlier protocol of Magyarik and Wagner [2] was *not* based on non-recursiveness of a decision problem, contrary to what the title of their paper may suggest; this was recently pointed out in [1]). One of the problems with the paper [6] is that it is somewhat too heavy on combinatorial group theory, at least for a non-expert. Most of that group theory is needed to separate the receiver (Alice) and the adversary (Eve) in power. This is, indeed, a very non-trivial problem that opens several interesting research avenues.

Here our primary focus is on the "encryption emulation" attack on the sender's (Bob's) transmissions. We show that Bob's encryption can be made secure against the "encryption emulation" attack by computationally unbounded adversary, with one reservation: a legitimate receiver decrypts correctly with probability that can be made arbitrarily close to 1, but not equal to 1.

First we recall what the "encryption emulation" attack on the sender's encryption is:

> Eve emulates the encryption algorithm over and over again, each time with fresh randomness, until the transmission to be attacked is obtained; this will happen eventually with overwhelming probability. The correctness of the scheme then guarantees that the corresponding secret key (as obtained by the adversary performing key generation) allows to decrypt illegitimately.

This attack would indeed work fine (at least, for computationally unbounded adversary) if the correctness of the scheme was perfect. However, if there is a gap, no matter how small (it can be easily made on the order of 10^{-200}, see [6]), between 1 and the probability of correct decryption by a legitimate receiver, then this gap can be very substantially "amplified" for the adversary, thus making the probability of correct illegitimate decryption anything but overwhelming. To explain how and why this is possible, we do not really need to introduce any serious group theory.

We emphasize at this point that when we talk about security against "computationally unbounded adversary" in this paper, we do not claim security against the "encryption emulation" (or any other) attack on the receiver's public key or decryption algorithm, but we only claim security against the encryption emulation attack *on the sender's transmission*. It seems that the problem of security of the sender's encryption algorithm is of independent interest. Of course, in commercial applications to, say, Internet shopping or banking, both the sender's and the receiver's algorithms are assumed to be known to the adversary ("Kerckhoffs' assumptions"), and the receiver's decryption algorithms (or algorithms for obtaining public keys) are usually more vulnerable to attacks. However, in some other applications, say, to electronic signatures (not to mention non-commercial, e.g. military applications), decryption algorithms or algorithms for generating public keys (by the receiver) need not be public, whereas encryption algorithms (of the sender) always are. It is therefore important to have a consensus in the cryptographic community on the first of the two security claims in Section 1 of the present paper.

Thus, in Section 3, we present a protocol which is secure against the encryption emulation attack on the sender's transmission by a computationally unbounded adversary who has complete

information on the algorithm(s) and hardware that the sender uses for encryption. More precisely, in our protocol the sender transmits his private bit sequence by encrypting one bit at a time, and the receiver decrypts each bit correctly with probability that can be made arbitrarily close to 1, but not equal to 1. *At the same time, the (computationally unbounded) adversary decrypts each bit (by emulating the sender's encryption algorithm) correctly with probability at most* $\frac{3}{4}$.

There are essentially no requirements on the sender's computational abilities; in fact, encryption can be done by hand, which can be a big advantage in some situations; for example, a field operative can receive a public key from a command center and transmit encrypted information over the phone, without even using a computer.

In Section 5, we use the same protocol to address the second claim from our Section 1; in particular, we allow the adversary to emulate both the encryption and decryption algorithms. More precisely, here we allow the adversary to emulate the receiver's algorithm for obtaining the public key. If such an attack is successful, the adversary recovers the receiver's private key used for decryption, and then the encryption emulation attack on the sender's encryption will allow the adversary to decrypt correctly with the same probability as the receiver would. We show however that there is an algorithm for the receiver to obtain her public key that takes time $O(k^3)$ (where k is the complexity of the public key), whereas the adversary who would like to emulate this algorithm with all possible randomness would have to take time $O(k^k)$.

3. Encryption protocol

In this section, we describe an encryption protocol with the following features:

(F1) Bob encrypts his secret bit by a word in a public alphabet X.

(F2) Alice (the receiver) decrypts Bob's transmission correctly with probability that can be made arbitrarily close to 1, but not equal to 1.

(F3) The adversary, Eve, is assumed to have no bound on the speed of computation or on the storage space.

(F4) Eve is assumed to have complete information on the algorithm(s) and hardware that Bob uses for encryption. However, Eve cannot predict outputs of Bob's random numbers generator (the latter could be just coin tossing, say). Neither does she know Alice's algorithm for obtaining public keys.

(F5) Eve cannot decrypt Bob's secret bit correctly with probability $> \frac{3}{4}$ by emulating Bob's encryption algorithm.

Once again: in this section, we only claim security against the "encryption emulation" attack (by computationally unbounded adversary) on the sender's transmissions. This does not mean that the receiver's private keys in our protocol are insecure against real-life (i.e., computationally bounded) adversaries, but this is the subject of Section 5. Here we prefer to focus on what is secure against computationally unbounded adversary since this paradigm shift looks important to us (at least, from the theoretical point of view).

We also have to reiterate that the encryption protocol which is presented in this section is probably not very suitable for commercial applications (such as Internet shopping or banking) due to a large amount of work required from Alice to receive just one bit from Bob. Bob, on the other hand, may not even need a computer for encryption.

Now we are getting to the protocol description. In one round of this protocol, Bob transmits a single bit, i.e., Alice generates a new public key for each bit transmission.

(P0) Alice publishes two group presentations by generators and defining relators:

$$\Gamma_1 = \langle x_1, x_2, \ldots, x_n \mid r_1, r_2, \ldots, r_k \rangle$$

$$\Gamma_2 = \langle x_1, x_2, \ldots, x_n \mid s_1, s_2, \ldots, s_m \rangle.$$

One of them defines the trivial group, whereas the other one defines an infinite group, but only Alice knows which one is which. In the group that is infinite, Alice should be able to efficiently solve the word problem, i.e., given a word $w = w(x_1, x_2, \ldots, x_n)$, she should be able to determine whether or not $w = 1$ in that group. There is a large and easily accessible pool of such groups (called *small cancellation groups*), see [6] for discussion.

Bob is instructed to transmit his private bit to Alice as follows:

(P1) In place of "1", Bob transmits a pair of words (w_1, w_2) in the alphabet $X = \{x_1, x_2, \ldots, x_n, x_1^{-1}, \ldots, x_n^{-1}\}$, where w_1 is selected randomly, while w_2 is selected to be equal to 1 in the group G_2 defined by Γ_2.

(P2) In place of "0", Bob transmits a pair of words (w_1, w_2), where w_2 is selected randomly, while w_1 is selected to be equal to 1 in the group G_1 defined by Γ_1.

Now we have to specify the algorithms that Bob should use to select his words.

Algorithm "0" (for selecting a word $v = v(x_1, \ldots, x_n)$ not equal to 1 in a Γ_i) is quite simple: Bob just selects a random word by building it letter-by-letter, selecting each letter uniformly from the set $X = \{x_1, \ldots, x_n, x_1^{-1}, \ldots, x_n^{-1}\}$. The length of such a word should be a random integer from an interval that Bob selects up front, based on his computational abilities. In the end, Bob should cancel out all subwords of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$.

Algorithm "1" (for selecting a word $u = u(x_1, \ldots, x_n)$ equal to 1 in a Γ_i) is slightly more complex. It amounts to applying a random sequence of operations of the following two kinds, starting with the empty word:

(1) Inserting into a random place in the current word a pair hh^{-1} for a random word h.
(2) Inserting into a random place in the current word a random conjugate $g^{-1} r_i g$ of a random defining relator r_i.

In the end, Bob should cancel out all subwords of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$. The length of the resulting word should be in the same range as the length of the output of Algorithm "0". We do not go into more details here because all claims in this section remain valid no matter what algorithm for producing words equal to 1 is chosen, as long as it returns a word whose length is in the same range as that of the output of Algorithm "0".

Now let us explain why the legitimate receiver (Alice) decrypts correctly with overwhelming probability. Suppose, without loss of generality, that the group G_1 is trivial, and G_2 is infinite. Then, if Alice receives a pair of words (w_1, w_2) such that $w_1 = 1$ in G_1 and $w_2 \neq 1$ in G_2, she concludes that Bob intended to transmit a "0". This conclusion is correct with probability 1. If Alice receives (w_1, w_2) such that $w_1 = 1$ in G_1 and $w_2 = 1$ in G_2, she concludes that Bob intended to transmit a "1". This conclusion is correct with probability which is close to 1, but not equal to 1 because it may happen, with probability $\epsilon > 0$, that the random word w_2 selected by Bob is equal to 1 in G_2. The point here is that, if G_2 is infinite, this ϵ is negligible and, moreover, for "most" groups G_2 this ϵ tends to 0 exponentially fast as the length of w_2 increases. For more precise statements, see [6]; here we just say that it is easy for Alice to make sure that G_2 is one of those groups.

4. Emulating encryption

Now we are going to discuss Eve's attack on Bob's transmission. Under our assumptions (F3), (F4) Eve can identify the word(s) in the transmitted pair which is/are equal to 1 in the corresponding group(s), as well as the word, if any, which is not equal to 1. Indeed, for any particular transmitted word w she can use the "encryption emulation" attack, as described in our Introduction: she emulates algorithms '0" and "1" over and over again, each time with fresh randomness, until the word w is obtained. Thus, Eve is building up two lists, corresponding to two algorithms above. Our first observation is that the list that corresponds to the Algorithm "0" is useless to Eve because it is eventually going to contain *all* words in the alphabet $X = \{x_1, \dots, x_n, x_1^{-1}, \dots, x_n^{-1}\}$, with overwhelming probability. Therefore, Eve may just as well forget about this list and concentrate on the other one, that corresponds to the Algorithm "1". Now the situation boils down to the following: if the word w appears on the list, then it is equal to 1 in the corresponding group G_i. If not, then not.

It may seem that Eve should encounter a problem detecting $w \neq 1$: how can she conclude that w does *not* appear on the list if the list is infinite (more precisely, of a priori unbounded length) ? This is where our condition (F4) plays a role: if Eve has complete information on the algorithm(s) and hardware that Bob uses for encryption, then she does know a bound on the size of the list.

Thus, Eve can identify the word(s) in the transmitted pair which is/are equal to 1 in the corresponding group(s), as well as the word, if any, which is not equal to 1. There are the following possibilities now:

(1) $w_1 = 1$ in G_1, $w_2 = 1$ in G_2;
(2) $w_1 = 1$ in G_1, $w_2 \neq 1$ in G_2;
(3) $w_1 \neq 1$ in G_1, $w_2 = 1$ in G_2.

It is easy to see that one of the possibilities (2) or (3) cannot actually occur, depending on which group G_i is trivial. Then, the possibility (1) occurs with probability $\frac{1}{2}$ (either when Bob wants to transmit "1" and G_1 is trivial, or when Bob wants to transmit "0" and G_2 is trivial). If this possibility occurs, Eve cannot decrypt Bob's bit correctly with probability $> \frac{1}{2}$ because she does not know which group G_i is trivial. As we have pointed out in the Introduction, a computationally unbounded Eve could find out which G_i is trivial, but we specifically consider attacks on the sender's encryption in this paper. We just note, in passing, that for a real-life (i.e., computationally bounded) adversary to find out which presentation Γ_i defines the trivial group is by no means easy and deserves to be a subject of separate investigation. There are many different ways to efficiently construct very complex presentations of the trivial group, some of them involving a lot of random choices. See e.g. [5] for a survey on the subject.

In any case, our claim (F5) was that Eve cannot decrypt Bob's bit correctly with probability $> \frac{3}{4}$ by emulating Bob's encryption algorithm, which is obviously true in this scheme since the probability for Eve to decrypt correctly is, in fact, precisely $\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot 1 = \frac{3}{4}$. (Note that Eve decrypts correctly with probability 1 if either of the possibilities (2) or (3) above occurs.)

Someone may say that $\frac{3}{4}$ is a rather high probability of illegitimate decryption, even though this is just for one bit. Recall however that we are dealing with computationally unbounded adversary, while Bob can essentially do his encryption by hand! All he needs is a generator of uniformly distributed random integers in the interval between 1 and $2n$ (the latter is the cardinality of the alphabet X). Besides, note that with the probability of correctly decrypting one bit equal to $\frac{3}{4}$, the probability to correctly decrypt, say, a credit card number of 16 decimal digits would be on the order of 10^{-7}, which is comparable to the chance of winning the jackpot in a lottery. Of course,

there are many tricks that can make this probability much smaller, but we think we better stop here because, as we have pointed out before, our focus here is on the new paradigm itself.

5. Encryption/decryption emulation attack by a computationally superior yet bounded adversary

In this section, we show that Eve would need a serious computational power (superexponential compared to that of Alice) to run an emulation attack on Alice's algorithm for generating a public key if this algorithm is sophisticated enough. This attack is similar to the emulation attack on Bob's encryption that we considered before:

> Eve emulates Alice's algorithm for generating a public key over and over again, each time with fresh randomness, until the actual public key is obtained; this will happen eventually with overwhelming probability.

Obviously, if this attack is successful, then the encryption emulation attack on the sender's encryption, as described in the Introduction, will allow Eve to decrypt Bob's bit correctly with overwhelming probability because Eve would know, just like Alice does, which public presentation Γ_i is a presentation of the trivial group.

Thus, we are going to focus on the above attack and describe a particular algorithm that Alice can use to generate a presentation of the trivial group, such that emulating this algorithm with all possible randomness would entail going over superexponentially many possibilities.

The algorithm itself is quite simple, and it produces special kinds of presentations of the trivial group. It is known (see e.g. [4], [5]) that the following presentations define the trivial group:

$$\langle x, y|\ x^{-1}y^n x = y^{n+1},\ w = 1\rangle,$$

where $n \geq 1$ and w is any word in x and y with exponent sum 1 on x. Let us assume that the length of w is k, a sufficiently large integer selected by Alice, which can be considered a measure of complexity of the above presentation. Technically, the integer n, too, influences complexity of the presentation, but it is less important to us, so we will consider n fixed (and rather small) in what follows.

Now we are going to describe a k-step algorithm that Alice can use to obtain a presentation of complexity $O(k^3)$ that would define the trivial group.

(1) At the first step, Alice selects a presentation of the form

$$\langle x_1, y_1|\ x_1^{-1}y_1^n x_1 = y_1^{n+1},\ w_1 = 1\rangle,$$

with a random word w_1 in x_1 and y_1 of length k, having exponent sum 1 on x_1.

(2) At the ith step, $1 < i < k$, Alice adds two new generators, x_i and y_i, and a new relator w_i, which is a random word in x_i and y_i of length k, having exponent sum 1 on x_i. Also, she adds the relation $x_i^{-1}y_i^n x_i = y_i^{n+1}$. The resulting presentation still defines the trivial group; in particular, $x_i = 1$ and $y_i = 1$ in this group. After that, Alice "mixes" new generators with old ones by inserting $x_i^{\pm 1}$ and $y_i^{\pm 1}$ in k random places in each old relator. The idea is to have roughly k generators with index i in each old relator.

(3) The kth step is special. Alice adds two new generators, x_k and y_k, and two relators, $x_k^{-1}y_k x_k y_k^{-2}$ and w_k of small length (say, between 3 and 6) having exponent sum 1 on x_k. The resulting group is therefore still trivial; in particular, all generators are equal to 1 in this group. Then Alice adds roughly k more relators to this presentation, where each relator is a random word of length 1, 2, or 3 in the generators $x_1, y_1, \ldots, x_k, y_k$. The only thing Alice takes care of is that each generator occurs in at least one of these new relators. Then Alice

mixes these new relators with each other and with the two relators $x_k^{-1}y_kx_ky_k^{-2}$ and w_k by using operations of the following kinds: $r_i \to r_ir_j$, or $r_i \to r_ir_j^{-1}$, or $r_i \to r_jr_i$, or $r_i \to r_j^{-1}r_i$ for different relators r_i, r_j, until all relators will have roughly k occurrences of each generator.

(4) Finally, Alice randomly renames the generators to hide the order of steps.

It is fairly obvious that the complexity of the resulting presentation is $O(k^3)$, whereas to emulate all possible randomness in the above algorithm Eve would have to take time $O(k^k)$ because she would have to explore, in particular, random (sub)words of length k on $2k$ letters (and their inverses).

6. A challenge

Here we offer a computational challenge to illustrate one of our points, namely, that Eve might need a serious computational power to detect a presentation of the trivial group among two given presentations.

Let

$$\Gamma_1 = \langle x, y, z \mid\ x^{-1}zy^{-1}x^{-1}yxzx^{-1}y^{-1}xyz^{-1},\ y^{-1}x^{-1}yxz^{-1}x^{-1}y^{-1}xyxzx^{-1}y^{-1}x,\ xy^{-1}z^2x^{-1}y^{-2}x^3z^{-1}x^{-1}y\rangle,$$

$$\Gamma_2 = \langle x, y, z \mid\ x^{-1}zy^{-1}x^{-1}yxzx^{-1}y^{-1}xyz^{-1},\ y^{-1}x^{-1}yxz^{-1}x^{-1}y^{-1}xyx^{-1}zxy^{-1}x,\ xy^{-1}z^2x^{-1}y^{-2}x^3z^{-1}x^{-1}y\rangle.$$

The question is: which Γ_i is a presentation of the trivial group?

References

[1] J.-C. Birget, S. Magliveras, M. Sramka, *On public-key cryptosystems based on combinatorial group theory*, Tatra Mountains Mathematical Publications **33** (2006), 137–148.

[2] M. R. Magyarik, N. R. Wagner, *A Public Key Cryptosystem Based on the Word Problem*, in CRYPTO 1984, Lecture Notes Comp. Sci. **196** (1985), 19–36.

[3] A. J. Menezes, *Handbook of Applied Cryptography*. CRC Press, 1996.

[4] C. F. Miller and P. Schupp, *Some presentations of the trivial group*, Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference on Geometric Group Theory and Computer Science. Contemp. Math., Amer. Math. Soc. **250** (1999), 113–115.

[5] A. D. Myasnikov, A. G. Myasnikov, V. Shpilrain, *On the Andrews-Curtis equivalence*, Contemp. Math., Amer. Math. Soc. **296** (2002), 183–198.

[6] V. Shpilrain and G. Zapata, *Using decision problems in public key cryptography*, preprint. http://www.sci.ccny.cuny.edu/~shpil/wppkc.pdf

Department of Mathematics, The City College of New York, New York, NY 10031

E-mail address: denis.osin@gmail.com, shpil@groups.sci.ccny.cuny.edu

http://www.sci.ccny.cuny.edu/~osin/

http://www.sci.ccny.cuny.edu/~shpil/