Guess what?! On the impossibility of unconditionally secure public-key encryption

Lorenz Panny*

Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands

Received: 3rd June 2020 | Revised: 10th September 2020 | Accepted: 10th September 2020

Abstract *We (once again) refute recurring claims about a public-key encryption scheme that allegedly provides unconditional security. This is approached from two angles: We give an information-theoretic proof of impossibility, as well as a concrete attack breaking the proposed scheme in essentially no time.*

Keywords: public-key cryptography, perfect secrecy, information theory, impossibility, cryptanalysis

2010 Mathematics Subject Classification: 94A60, 68P30, 68Q87

1 INTRODUCTION

In 2017, *Guess Again*, a public-key encryption scheme claiming *unconditional* security against passive eavesdroppers, was submitted to NIST's call for post-quantum cryptography [2]. Although we publicly broke that scheme with a fast attack script about three hours after the proposals were published by NIST [6], the authors still have not acknowledged the attack nor withdrawn their proposal (though NIST deselected it from advancing to the second round). About seven months later, a paper by a subset of the *Guess Again* authors describing essentially the same system, with no mention of our earlier attack, appeared at *ICMS 2018* [1]. The abstract of that paper states:

> 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 $1/2$.

In this note, we show that this claim is false, and in fact impossible to achieve.

Acknowledgements. I thank Tanja Lange and Andreas Hülsing for their helpful comments on an earlier draft. Thanks to the anonymous reviewers for their suggestions (including, but not limited to, Remark 3).

This work was supported in part by the Commission of the European Communities through the Horizon 2020 program under project number 643161 (ECRYPT-NET).

2 THEORY

A typical course in cryptography analyzes the one-time pad early on, deemed a helpful thing to study for multiple reasons: its tremendous historical significance, as a first basic (but illuminating) example of formally provable security, and to provide motivation and intuition for related concepts such as stream ciphers. Later in the cryptographic curriculum, public-key cryptography is introduced, and alas, it turns out that the absolute security guarantee of the one-time pad is unachievable in that setting — computational hardness assumptions must be made to separate honest users from attackers. This observation was pointed out as early as 1976, in the very paper in which Diffie and Hellman invented public-key cryptography [3]:

> We note that neither public key cryptosystems nor one-way authentication systems can be unconditionally secure because the public information always determines the secret information uniquely among the members of a finite set. With unlimited computation, the problem could therefore be solved by a straightforward search.

Notice that this well-known impossibility argument assumes a scheme in which each public key comes from at most one private key. This is the apparent "gap" in the impossibility proof that the construction of [2, 1] tries to exploit, by making the scheme probabilistic and incorporating decryption errors (i.e., occasionally decrypting correctly generated ciphertexts into messages different from the original plaintext). Thus, indeed, the public information does not always uniquely identify the secret information anymore and the reasoning above does not apply immediately. However, a similar brute-force argument still works in the presence of decryption errors: Instead of defining a single unique plaintext, a given ciphertext now determines a *likelihood distribution* on the set of potential messages, and the attacker can (using massive but finite computation) simply evaluate this likelihood function and output the *most likely* messages as their guess for the plaintext. Lemma 2 below shows that this attacker "never loses".

*Corresponding Author: lorenz@yx7.cc

Remark 1. *Another viewpoint to approach the question of unconditional public-key cryptography was pursued by Maurer in the nineties [4]: Using information-theoretical bounds on conditional entropy and mutual information, he concludes that two parties cannot possibly negotiate a shared secret over a public channel in an unconditionally secure manner:*

> *[...] if Alice and Bob do not share at least some partially secret information initially, they cannot generate an information-theoretically secure secret key S [...] if they can only communicate over a public channel accessible to Eve, even if this channel is authenticated.*
>
> *This fact can be rephrased as follows: There exists no unconditionally-secure public-key cryptosystem or public-key distribution protocol.* *[Footnote]*

While, indeed, the machinery from [4] is certainly much stronger than necessary to imply Lemma 2, we will for simplicity present a more concrete and computational proof similar to the brute-force argument from [3] quoted above.

2.1 PROOF OF IMPOSSIBILITY

Recall the protocol flow of a public-key encryption scheme:

- Bob generates a key pair $(\mathsf{pk}, \mathsf{sk})$ and publishes the public key pk.
- Alice encrypts a message m and sends the ciphertext $c = \mathsf{Enc}_{\mathsf{pk}}(m)$ to Bob.
- Bob decrypts c to obtain a message $m' = \mathsf{Dec}_{\mathsf{sk}}(c)$.

Here, $\mathsf{Enc}_{\mathsf{pk}}()$ and $\mathsf{Dec}_{\mathsf{sk}}()$ are both probabilistic algorithms. Note that it is not required that any of the involved procedures be efficient; in fact, in the following, we only have to assume that $\mathsf{Enc}_{\mathsf{pk}}$ will *eventually* halt almost surely, i.e., with probability 1, for all messages.

Definition 1. *Consider a public-key encryption scheme with the interface above and message space* $\{0,1\}$*, and fix a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*

For any algorithm $\mathcal{O}$ *taking a ciphertext and returning a single bit, we define the* distinguishing advantage

$$\mathrm{Adv}(\mathcal{O}) := \Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(1)) = 1] - \Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(0)) = 1]\,,$$

where the probabilities are taken over the randomness consumed by $\mathcal{O}$ *and* $\mathsf{Enc}_{\mathsf{pk}}$*.*

Remark 2. *Typical definitions of the advantage take an absolute value to symmetrize the definition with respect to distinguishers which are* worse *than pure guessing — and are thus in fact* good *at distinguishing. In our scenario, this would clutter the notation and introduce unnecessary case distinctions, so we stick to the present definition and point out that a negative advantage can easily be turned into a positive one of equal magnitude by inverting the distinguisher's output.*

Also note that this definition of advantage equals the "other" choice $\Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(0)) = 0] - \Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(1)) = 0]$.

Lemma 1. *As before, consider a public-key encryption scheme with the interface above and message space* $\{0,1\}$*, fix a key pair* $(\mathsf{pk}, \mathsf{sk})$*, and let* $\mathcal{O}$ *be a (not necessarily deterministic or efficient) algorithm that takes as input a ciphertext and outputs a single bit.*

Then, writing $p_m(c) := \Pr[\mathsf{Enc}_{\mathsf{pk}}(m) = c]$ *for shorthand, it holds that*

$$\mathrm{Adv}(\mathcal{O}) \;\le\; \sum_c \max\left\{0,\, p_1(c) - p_0(c)\right\}.$$

Proof. We first unroll the definition of $\mathrm{Adv}(\mathcal{O})$, using linearity of expectations and the fact that $\Pr[X = 1] = E[X]$ when X is a random bit:

$$\begin{aligned}
\mathrm{Adv}(\mathcal{O}) &= \Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(1)) = 1] - \Pr[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(0)) = 1] \\
&= \mathrm{E}\big[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(1))\big] - \mathrm{E}\big[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(0))\big] \\
&= \mathrm{E}\big[\mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(1)) - \mathcal{O}(\mathsf{Enc}_{\mathsf{pk}}(0))\big] \\
&= \sum_c \big(\Pr[\mathsf{Enc}_{\mathsf{pk}}(1) = c] - \Pr[\mathsf{Enc}_{\mathsf{pk}}(0) = c]\big) \cdot \mathrm{E}[\mathcal{O}(c)] \\
&= \sum_c \big(p_1(c) - p_0(c)\big) \cdot \mathrm{E}[\mathcal{O}(c)]\,. \qquad (*)
\end{aligned}$$

There are positive and negative terms in the sum $(*)$, determined by the sign of $p_1(c) - p_0(c)$. To maximize the advantage, it is clearly optimal to maximize $\mathrm{E}[\mathcal{O}(c)]$ when $p_1(c) > p_0(c)$ and minimize it when $p_0(c) > p_1(c)$.

Since $O(c)$ is a random variable on $\{0,1\}$, its expectation $\mathrm{E}[O(c)]$ must lie in the interval [0;1]; therefore, the value of $\mathrm{Adv}(O)$ is upper bounded by the quantity

$$\sum_c \big(p_1(c) - p_0(c)\big) \cdot \begin{cases} 0 & \text{if } p_0(c) \geq p_1(c); \\ 1 & \text{if } p_1(c) > p_0(c). \end{cases}$$

This is the same thing as

$$\sum_c \max\big\{0,\, p_1(c) - p_0(c)\big\}\,. \qquad \square$$

Example. *For a public-key encryption scheme on* $\{0,1\}$ *with guaranteed correct decryption,* $\mathsf{Dec}_{\mathsf{sk}}$ *has advantage one: Since* $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m$ *for all* m*, the definition immediately simplifies to* $\mathrm{Adv}(\mathsf{Dec}_{\mathsf{sk}}) = \Pr[1 = 1] - \Pr[0 = 1] = 1$.

On the other hand, a distinguisher O *that ignores its input and simply outputs a uniformly random bit has advantage zero, since both probabilities in the definition of* $\mathrm{Adv}(O)$ *equal* $1/2$ *in that case.*

Lemma 2. *As above, consider a public-key encryption scheme with message space* $\{0,1\}$*,*[1] *and fix a public key* $(\mathsf{pk}, \mathsf{sk})$.

Using $\mathsf{Enc}_{\mathsf{pk}}$*, one can construct a deterministic (but not necessarily efficient) algorithm* $\mathcal{A}_{\mathsf{pk}}$ *that achieves the upper bound proved in Lemma 1. In particular, it is at least as good as* $\mathsf{Dec}_{\mathsf{sk}}$ *at distinguishing the two random variables* $\mathsf{Enc}_{\mathsf{pk}}(0)$ *and* $\mathsf{Enc}_{\mathsf{pk}}(1)$*, despite not knowing the secret key.*

Proof. The argument consists of two parts: Constructing a maximum-likelihood estimator $\mathcal{A}_{\mathsf{pk}}$ for the message m from (a single sample of) the random variable $\mathsf{Enc}_{\mathsf{pk}}(m)$, and then showing that $\mathcal{A}_{\mathsf{pk}}$ achieves the upper bound from Lemma 1 to conclude that $\mathsf{Dec}_{\mathsf{sk}}$ cannot possibly do better than $\mathcal{A}_{\mathsf{pk}}$.

The adversary $\mathcal{A}_{\mathsf{pk}}$ is constructed as follows: Given a ciphertext c, compute the two probabilities $p_0(c) = \Pr[\mathsf{Enc}_{\mathsf{pk}}(0) = c]$ and $p_1(c) = \Pr[\mathsf{Enc}_{\mathsf{pk}}(1) = c]$ by iterating through all possible values of the randomness consumed by $\mathsf{Enc}_{\mathsf{pk}}$ and counting how often the ciphertext c is observed for either message. Then simply output the plaintext guess 0 if $p_0(c) \geq p_1(c)$ and the guess 1 if $p_1(c) > p_0(c)$. Via the assumption that $\mathsf{Enc}_{\mathsf{pk}}$ halts almost surely, it consumes only a finite amount of randomness almost surely, hence $\mathcal{A}_{\mathsf{pk}}$ can within finitely many computational steps approximate $p_0(c)$ and $p_1(c)$ well enough to determine which one is bigger. Notice that the adversary $\mathcal{A}_{\mathsf{pk}}$ indeed implements a maximum-likelihood estimator for the message $m \in \{0,1\}$ from a single encryption $\mathsf{Enc}_{\mathsf{pk}}(m)$ of m. Moreover, the algorithm $\mathcal{A}_{\mathsf{pk}}$ achieves the upper bound from Lemma 1 (in fact, it is crafted precisely to do so), hence no $\mathsf{Dec}_{\mathsf{sk}}$ can ever do better than $\mathcal{A}_{\mathsf{pk}}$. $\square$

The claims from [2, 1] amount to the assertion that knowledge of the secret key permits honest users to succeed in decrypting with probability

$$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m] > 1/2\,,$$

while at the same time, for *all* adversaries $\mathcal{A}_{\mathsf{pk}}$ given only the public key,

$$\Pr[\mathcal{A}_{\mathsf{pk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m] = 1/2\,.$$

These statements imply $\mathrm{Adv}(\mathsf{Dec}_{\mathsf{sk}}) > 0$ and $\mathrm{Adv}(\mathcal{A}_{\mathsf{pk}}) = 0$, which is impossible due to Lemma 2. Essentially, the only way to make sure $\mathcal{A}_{\mathsf{pk}}$ cannot do better than guessing is to transmit no information whatsoever about the plaintext in the ciphertext, but then $\mathsf{Dec}_{\mathsf{sk}}$ must also resort to plainly guessing what the encrypted message was supposed to be, which is utterly useless; in fact, it entirely defeats the purpose of communication as defined by Shannon in his seminal paper [7]: *"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."*

Remark 3. *The contents of this section can equivalently be phrased in terms of statistical distances: The absolute value of* $\mathrm{Adv}(O)$ *is the distance between the output distributions of* O *when given encryptions of either* 0 *or* 1*. Lemma 1 is the probability preservation property. Lemma 2 is the result that for any two distributions there exists an optimal distinguisher, i.e., one that achieves the information-theoretic bound given by the statistical distance.*

3 PRACTICE

We now show how the concrete proposal *Guess Again* works, how it is broken in practice, and where exactly the flaw in the authors' security argument lies.

[1] The restriction to one-bit messages does not sacrifice any generality: Any message space of cardinality at least two can represent single bits, and conversely, bit strings suffice to encode any other message space.

Table 1: Biases in a depending on the secret random choices during encryption. Notice in particular that the decision whether the message bit is flipped or not is correlated with the bias in a.

$A \lessgtr B$	$t \leftarrow \{f,g\}$	bias in a	c
$<$	f	slightly biased towards B	m
$<$	g	strongly biased towards B	$1 \oplus m$
$>$	f	no significant bias	$1 \oplus m$
$>$	g	slightly biased away from B	m

We stress that demonstrating a concrete *efficient* attack is still relevant even when it is already known that information-theoretical security is impossible: The attacker from the proof is inefficient, and it could happen that in fact *all* attacks are inefficient, as is conjectured for some existing encryption schemes.

Note that the newer paper [1] does not use the name "Guess Again", nor does it mention the NIST process [5]. Since the proposals in [2] and [1] are essentially identical, we shall use the name *Guess Again* to refer to both, and point out differences where relevant.

3.1 DESCRIPTION OF *GUESS AGAIN*

The *Guess Again* scheme makes use of random walks. For each $x \in \mathbb{Z}$ and $k \geq 0$, we thus define the random variable $walk(x,k)$ as the end point of a random walk of k steps, each adding either $+1$ or -1 with probability $1/2$, starting from x. Write $s \leftarrow S$ for sampling s uniformly at random from a finite set S.

- Parameters: Positive integer constants n, f, g, h. [1] recommends $n = 256$, $g = h = 2000$, and $f = 100\,000$. Note [2] suggested $f = 120\,000$ instead.
- Key generation: Bob picks a private key $b \leftarrow \{0,...,n-1\}$ and computes the public key $B := walk(b,h)$. If $B \geq n-1$, Bob starts over with a new b.
- Encryption: Alice samples values $(a,t,\delta) \leftarrow \{B,...,n-1\} \times \{f,g\} \times \{\pm\frac{1}{2}\}$ and computes $A := walk(a,t) + \delta$. She repeats this step "sufficiently often" and groups the resulting pairs (a,A) into four lists $L_f^<$, $L_g^<$, $L_f^>$, and $L_g^>$, according to the condition $A \lessgtr B$ and the value of t.

  She then selects $r \leftarrow \{<,>\}$ and $s \leftarrow \{f,g\}$, and a pair $(a_0,A_0) \leftarrow L_s^r$. Her ciphertext equals (m,a_0) if $(r,s) \in \{(<,f),(>,g)\}$ and $(1 \oplus m, a_0)$ else.[2]

  Upon receiving such a ciphertext (c,a_0), Bob outputs either the bit $m' = c$ when $b < a_0$, or $m' = 1 \oplus c$ when $b \geq a_0$.

It is stated in [1, Section 6] that (for the example parameters above) this protocol has a ≈ 0.55 chance of transmitting a bit correctly, i.e., achieving $m' = m$.

3.2 BIASES IN THE CIPHERTEXTS

The security argument in [1, Section 4.2] states that the bit m is flipped with probability $1/2$ during the encryption procedure, hence c contains no information whatsoever about the plaintext bit m. Moreover, the additional component a_0 of the ciphertext is uniform in a public range $\{B,...,n-1\}$, hence also does not leak information about m. Probabilistically speaking, these observations can be phrased as the (indeed true) fact that the distributions of the values c and a_0 are both independent of the plaintext m.

However, the crucial flaw in the security argument of *Guess Again* is that **the *joint* distribution of (c,a_0) is *not* independent of m**. To see this, notice that those values of $a \leftarrow \{B,...,n-1\}$ that lead to $A < B$ must, on average, be closer to B than those that lead to $A > B$. Moreover, this tendency clearly depends on the chosen number of steps of the random walk: Shorter walks lead to closer nodes; in fact, the expected (absolute) distance travelled by a simple random walk of k steps is $\Theta(\sqrt{k})$. Therefore, there is a probabilistic dependency between the choice of $t \in \{f,g\}$ and the condition $A \lessgtr B$. Working out the distribution of a conditioned on the choice of $A \lessgtr B$ and $t \leftarrow \{f,g\}$ is not particularly difficult, but rather tedious, hence we only give qualitative simulation results showing the distribution of a relative to each of the four choices. See Figure 1 and Table 1. The bottom line is that *"flipped" ciphertexts are smaller on average,* which is precisely the leakage exploited by the (simple) attack script in Section 3.3.

[2] The bit-flipping description deviates from [2, 1], which uses the equivalent viewpoint of labelling intervals instead.

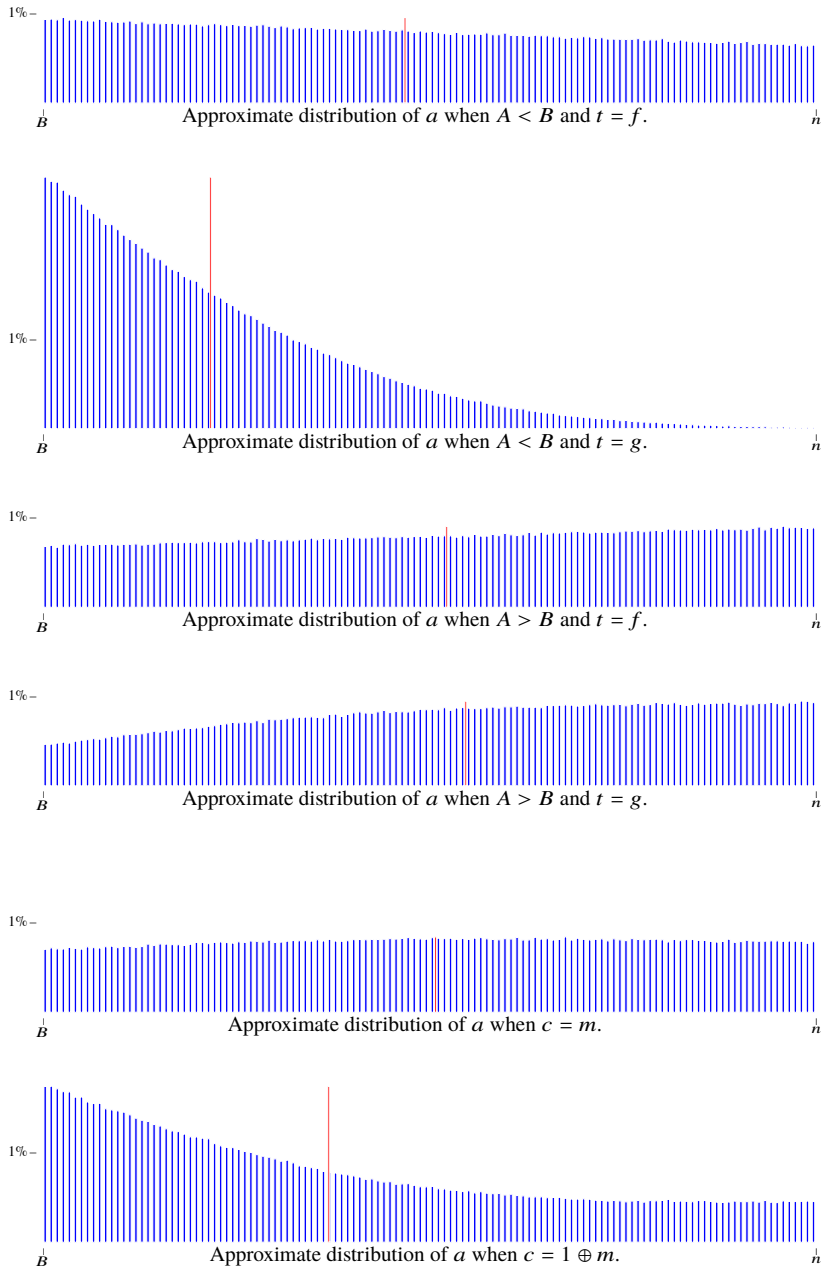

Figure 1: Approximate distribution of a conditioned on the property $A \lesseqgtr B$ and the number of steps t, where $(n, f, g) = (256, 100\,000, 2000)$ and B has been fixed as $n/2$. (Similar graphs result from other public keys.) The mean of each distribution is marked by a red vertical line. For efficiency reasons, we approximated the distribution of the random walks $walk(x, k)$ by the normal distribution $\mathcal{N}(x, k)$. Each of these histograms contains one million data points.

Algorithm 1: Breaking *Guess Again* with simple statistical analysis.

Input: a list of *Guess Again* ciphertexts $\left[(c^{(1)}, a_0^{(1)}), ..., (c^{(\ell)}, a_0^{(\ell)})\right]$ encrypting the same bit $m \in \{0,1\}$, such that $\{c^{(i)} \mid i \in \{1,...,\ell\}\} = \{0,1\}$.

Output: the bit m, correct with high probability when ℓ is large.

Initialize empty lists X and Y.

for $i \in \{1,...,\ell\}$ **do**

  if $c^{(i)} = 0$ **then** append $a_0^{(i)}$ to X, **else** append $a_0^{(i)}$ to Y.

Compute the average values $x := \sum X / |X|$ and $y := \sum Y / |Y|$.

if $x \geq y$ **then return** 0 **else return** 1.

```
l = 4000

def recover_bit(ct, bit):
    assert bit < len(ct) // l
    ts = [struct.unpack('BB', ct[i:i+2]) for i in range(l*bit, l*(bit+1), 2)]
    xs, ys = [a for a, b in ts if b == 1], [a for a, b in ts if b == 2]
    return sum(xs) / len(xs) >= sum(ys) / len(ys)

def decrypt(ct):
    res = sum(recover_bit(ct, b) << b for b in range(len(ct) // l))
    return int.to_bytes(res, len(ct) // l // 8, 'little')
```

Figure 2: Excerpt of the attack script sent to NIST's mailing list [6].

3.3 THE ATTACK

Recall from Section 3.1. that a *Guess Again* ciphertext consists of a tuple (c, a_0) with c a single bit and a_0 a value between B and $n-1$. The bit c is either equal to the message bit m, or it is negated. In Section 3.2, we showed that the choice between $c = m$ and $c = 1 \oplus m$ influences the distribution of the corresponding value a_0; concretely, a_0 is biased towards B when $c = 1 \oplus m$. This means we can mount a distinguishing attack by comparing a_0 to its expected values relative to the assumptions $c = m$ and $c = 1 \oplus m$, and output a guess according to which distribution a_0 is more likely to come from. This distinguisher has a better than $1/2$ chance of identifying the message given a ciphertext.

When more than one ciphertext encrypting the same message is sent at a time to boost the success rate, as was done in [2], accumulating this leakage allows for a very reliable distinguisher (Algorithm 1): Group the ciphertexts by their bit c, compute the average value of a_0 in each group, and compare. Output the c with the bigger average a_0 as a guess for m. This approach is implemented in the script below and works on 100 % of the example ciphertexts included in [2].

In [1, Section 5], the authors — perhaps in response to, but with no mention of, our earlier attack — warn against naïvely encrypting the same bit multiple times, foreshadowing that "statistical attacks [...] may be possible". However, if the construction truly were information-theoretically secure, any number of repetitions would never leak information (assuming uncorrelated randomness), hence acknowledging the reality of "statistical attacks" is clearly incompatible with their claim of unconditional security upheld elsewhere in the paper.

Algorithm 1 gives a straightforward method to exploit this leakage. More practically, Figure 2 shows the essential part of our Python implementation [6] which was sent to NIST's `pqc-forum` mailing list on December 21st, 2017, not including boilerplate code to parse and break the example ciphertext files contained in *Guess Again*'s NIST submission package.

Note that this script does *not* implement the adversary from Lemma 2; it does not brute-force anything and in fact runs in time linear in the input size. We remark that it is certainly possible to tweak *Guess Again* in such a way that any concrete attack would be (much) less efficient, but the fundamental impossibility of *information-theoretic* security remains. Therefore, we conclude that the security claims of *Guess Again* are both theoretically impossible as well as demonstrably broken in practice.

REFERENCES

[1] Mariya Bessonov, Dima Grigoriev and Vladimir Shpilrain. "A Framework for Unconditionally Secure Public-Key Encryption (with Possible Decryption Errors)". In: *Mathematical Software – ICMS 2018*. Ed. by James H. Davenport, Manuel Kauers, George Labahn and Josef Urban. Springer, July 2018, pp. 45–54.

[2] Mariya Bessonov, Dima Grigoriev, Alexey Gribov and Vladimir Shpilrain. *Guess Again. Unconditionally secure public-key encryption (with possible decryption errors)*. Submission to [5]. 2017.

[3] Whitfield Diffie and Martin E. Hellman. "New directions in cryptography". In: *IEEE Trans. Information Theory* 22.6 (Nov. 1976), pp. 644–654.

[4] Ueli Maurer. "Information-Theoretic Cryptography". In: *Advances in Cryptology – CRYPTO '99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. `ftp://ftp.inf.ethz.ch/pub/crypto/publications/Maurer99.pdf`. Springer, Aug. 1999, pp. 47–64.

[5] National Institute of Standards and Technology. *Post-quantum cryptography standardization: Call for Proposals Announcement*. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization`. Dec. 2016.

[6] Lorenz Panny. *OFFICIAL COMMENT: Guess Again*. Message to the `pqc-forum@list.nist.gov` mailing list. `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/guess-again-official-comment.pdf`. 21 December 2017.

[7] Claude E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423.