**Hybrid Cryptography – Additional Notes**

**Introduction**

Firstly, it is important to note that the most general term to describe the world of codes and ciphers is in fact *cryptology.* There are two main divisions of cryptology, cryptography and cryptanalysis. Cryptography is concerned with the secure transmission of messages, so only the intended recipient can read it. Cryptanalysis on the other hand is concerned with attempting to break that secure transmission, so a third party can read the message without having knowledge of the key that encrypted it. Cryptography can be further broken down into three main fields, symmetric cryptography, asymmetric cryptography and the area of protocols, in which cryptosystems can be implemented securely in the real world, virtually always over the internet.

Cryptology is often thought to be a very modern subject, and while it has advanced hugely in the last few decades, it in fact dates to the time of Ancient Egypt. In more recent times, the German Enigma machine is a very well-known example of a cryptosystem, that was successfully broken during WW2 at Bletchley Park. In the 1970’s, modern day cryptography was born with the breakthrough in public key schemes, which was discovered firstly by a researcher at GCHQ (there is a plaque in GCHQ’s museum in their headquarters in Cheltenham to commemorate this by the way...), but this was classified up until 1997 so was not public knowledge. Looking to the future with quantum computing on the horizon, many of the systems we use today will be rendered useless, so research in this field is still an extremely active topic.



Image: GCHQ’s enigma machine from their museum (Source: BBC News)

**Caesar Cipher**

The Caesar Cipher is an incredibly simple and well-known cryptosystem, the first and often only cryptosystem most people know. To use it, the plaintext characters, which are often just from the standard alphabet, are all shifted along the alphabet by a set number of positions. For example, the message “abc” encrypted with a shift of 2 would produce the ciphertext of “cde”. It is named after Julius Caesar, who was known to make frequent use of it to secure messages relating to military movements.

The Caesar cipher generally uses the standard alphabet a, b, c, …. z. The key for it is the shift that has been applied, so is a number between 1 and 25 excluding the trivial shift. To decrypt an encrypted message, we simply perform the shift in reverse, so for the example above we shift all letters back in the alphabet by 2 positions, so the ciphertext “cde” decrypts to the plaintext “abc”.

The Caesar Cipher’s simplicity makes it incredibly easy to break. The simplest way to break it is by brute – force, so simply checking all possible shifts between 1 and 25 and seeing which one produces valid plaintext. This can be done in a millisecond on a computer and would only take a couple of minutes by hand. Another way to break it would be via letter – frequency analysis. The most common occurring letter in standard English is the letter “e”, so assuming the ciphertext is relatively long, we can match the most frequent letter in the ciphertext with the letter “e”, and there is a fair chance that the key for the message will be the shift needed to take “e” to the most frequent letter in the ciphertext. Either way, the Caesar Cipher is useless in practice!



Image: A standard Caesar cipher wheel used for quick encryption and decryption (Source: Wikipedia)

**Vigenère Cipher**

The Vigenère Cipher is a polyalphabetic cipher (a cipher using several substitution alphabets) that uses a series of Caesar Ciphers to encrypt a message, which is based on the letters of a known keyword. It was first developed in 1553 and managed to survive for 300 years before it was finally broken.

The first step in the encryption is establishing a keyword for the message. We should note that this does not need to be a word but should just be a series of alphabetic letters. Next, we write out this keyword above our plaintext message repeatedly until every plaintext character is covered. By converting the plaintext and the keyword characters to integers from 0 to 25 and adding each together in turn and taking the result mod 26, we can obtain our ciphertext as integers from 0 to 25. Simply convert this to letters in the usual sense to get the ciphertext as alphabetic letters.

To decrypt a message, simply convert the ciphertext letters back to numbers from 0 to 25 in the usual sense. Above each ciphertext number, write the numbers corresponding to the keyword repeatedly until all ciphertext has been covered. Subtract the keyword number from each ciphertext number, taking the result mod 26, and convert back to letters in the usual way. This will be our plaintext message again.

Although the Vigenère Cipher survived 300 years of being unbroken, it can be broken easily today. The main weakness with the cipher is the repeating keyword that is used repeatedly for the length of the message. If the keyword’s length can be correctly guessed, then all the ciphertext can be treated as a series of Caesar Ciphers which are simple to break individually. Thus, the Vigenère Cipher is not secure today either.

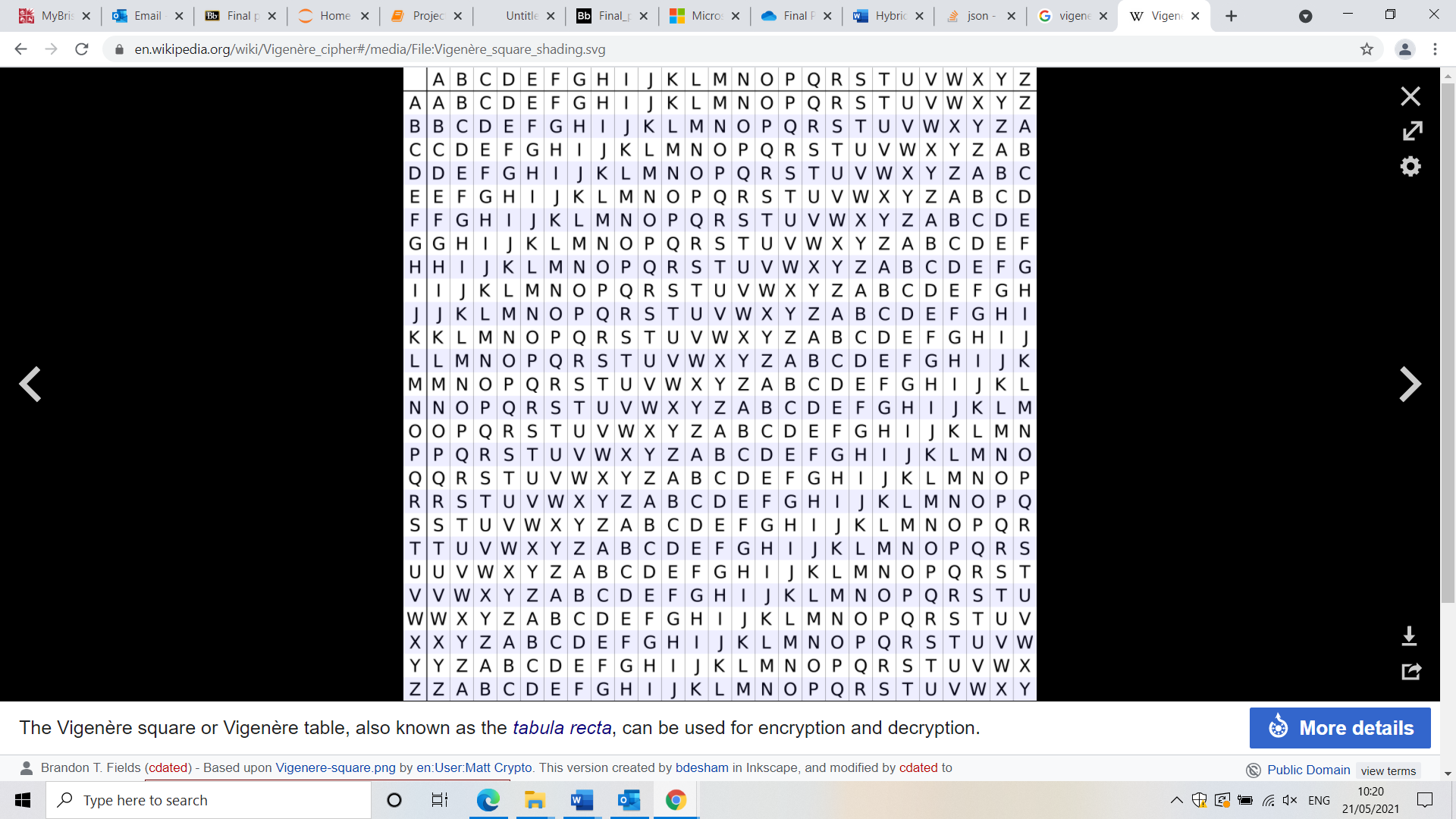


Image: A Vigenère Square, which is also knows as tabula recta, which can be used to fast encryption and decryption with the Vigenère cipher (Source: Wikipedia)

**RSA**

RSA is one of the most used public key schemes in existence today. It is used widely across cryptography for various applications, but in this project, we just consider its use in key transfer, so that is what I will discuss here, very briefly.

The idea behind RSA is that there are in fact two keys being used with the message, not just one like in symmetric cryptography. We have the public key, which is public knowledge, and the private key, which is private. The public key is distributed by the individual wishing to receive a message and is used by the sender to encrypt a message, and the private key (held only by the receiver) is used to decrypt it. Without knowing the private key, it is practically impossible to decrypt the message. When we say “message” here, we are nearly always referring to a symmetric key that is being transferred to another party to allow secure transmission of data using a symmetric cipher, commonly AES 128/192 or 3DES.

The fundamental principle that RSA relies on for its security is the (presumed, but not proved) difficulty of factoring the product of two very large primes. The reason for this is to find Euler’s totient function of this product, which is very easy to calculate once you know the factorization. From this, it would be a simple application of the Extended Euclidean Algorithm to arrive at the private key, d, which would decrypt the ciphertext and reveal the plaintext message, nearly always a key. Without being able to factorise the product, it is presumed very difficult to find Euler’s totient function of it, and thus incredibly difficult to find the private key d, giving RSA its security.

In practice today, each of the prime’s p, q should both be around 1024 bits long, giving an RSA modulus (N) of around 2048 bits. For now, this is more than enough security to defeat (most) attackers, but with the advancement in quantum computing a 2048-bit RSA key could be broken in a matter of hours, which will make the current algorithms and systems extremely vulnerable to attack.

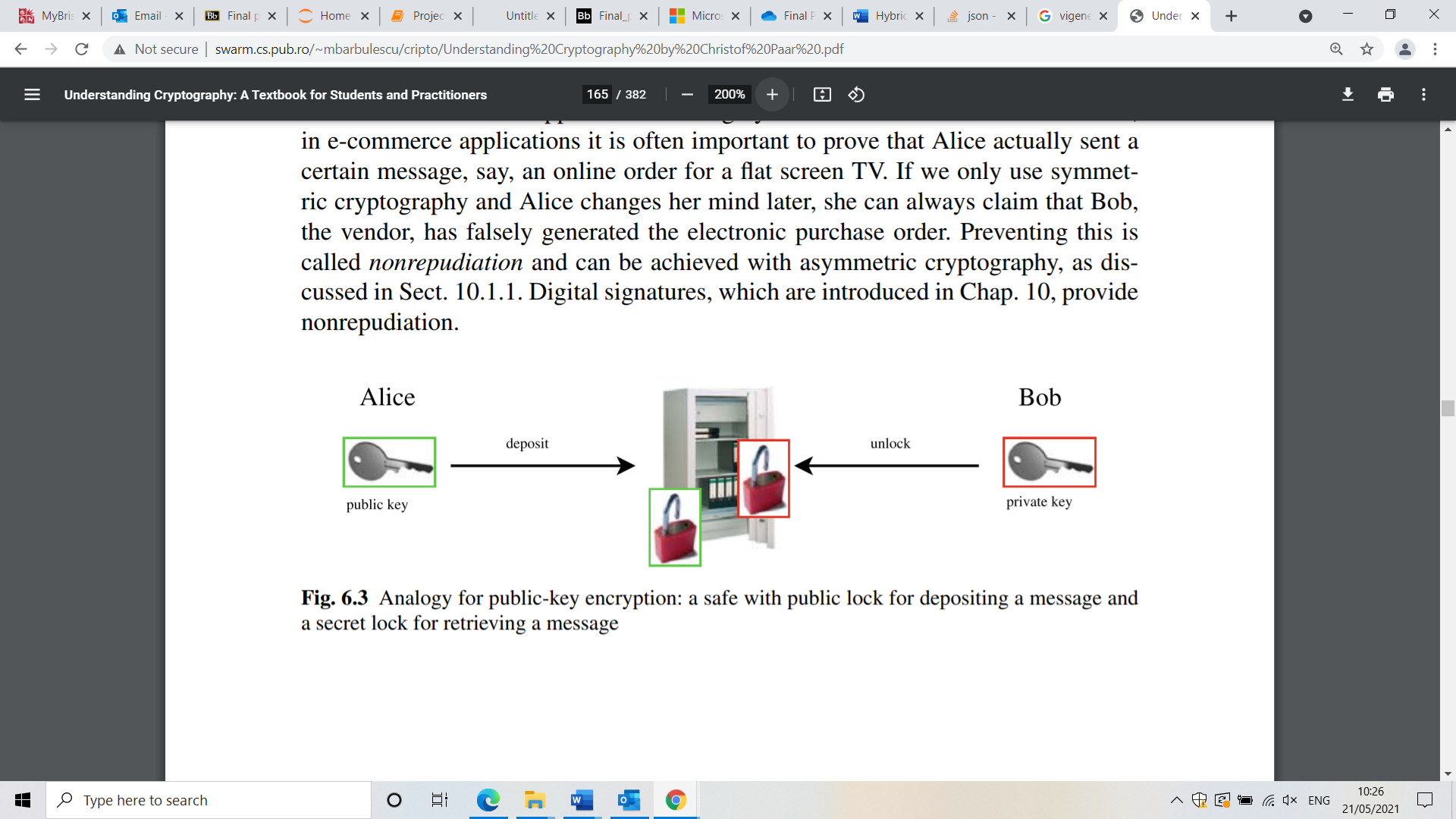


Image: A visual representation of how RSA works. Alice can use a public lock to deposit a message, and Bob uses a secret lock for retrieving the deposited message. (Source: Introduction to Cryptography, C.Paar Ruhr University Bochum, Germany)

**RSA – Related Algorithms**

Exponentiation is a crucial part of RSA in both the encryption and decryption stages, and with the size of numbers that are being used in the algorithm, it is crucial to have efficient methods for doing this. One common algorithm is the *square and multiply algorithm*.

The square and multiply algorithm is a very efficient method for exponentiation involving very large numbers. It gets its name from the operations it performs, squaring and multiplying. Firstly, the exponent we are dealing with is converted to binary, so becomes a string of ones and zeroes. The algorithm scans the exponent binary number from left to right. At each bit, it does the following:

1. For every exponent bit, the current result is squared.
2. If the current exponent bit is a 1, we perform a multiplication by the number we are raising to the exponent.
3. By repeating this for all the bits in the exponent, we get our final answer.

It is interesting to note that a convention when using RSA is to use specific exponents, with e = 65537 being the most common value to choose in practice. This has to do with a compromise between security and computational complexity, and the fact that as 65537 = 2^16 +1, so by the square and multiply algorithm we see that exponentiation is relatively straightforward, as there are few 1’s in the binary representation and many zero’s (so less multiplication)

A final point to make is that even though RSA is mathematically practically unbreakable, its practical implementation is just as important for its security in practice. Side channel attacks are attacks that exploit the implementation of an algorithm, rather than the algorithm itself. For example, this includes attacks focusing on the power consumption, timing and things such as sound that a computer could leak which could be exploited to extract the key. In RSA, a power trace could be used, which would result in an attacker seeing high and low voltage, which corresponds to the zero’s and one’s from the square and multiply algorithm. A long period of activity would indicate a bit value of 1 in the key (this requires squaring AND multiplication) and a period of low activity would indicate a bit value of 0 in the key (this just requires squaring). From this, the entire private key can be found.

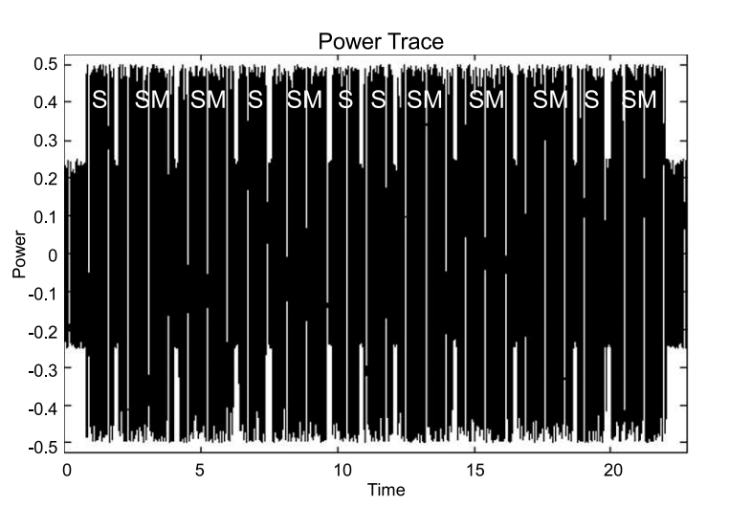


Image: Power trace of RSA (Source: Introduction to Cryptography, C.Paar Ruhr University Bochum, Germany)

**Kasiski Analysis**

We can systematically break Vigenère ciphertext with Kasiski analysis. The general method is to go through the message looking for repeated letter sequences. We then record the spacings between these sets of repeated sequences.

Say we have two noticeable repetitions – ‘abc’ four times and ‘def’ three times. Then, suppose the lengths between these are: 8, 24, 32, and 32, 8. Then, we can factorise these spacings and look at the most common factors. The factors with the highest count are the most likely lengths of the keyword.

Once we have a most likely length n, we extract every nth letter from the string, n times. For example, say the length was 4, then we extract every fourth letter starting with the first, then second, then third, and then fourth. This gives us four separate messages, but now each message is shifted by one letter only – they are Caesar shifted.

Lastly, we can go through each string and perform a frequency analysis of the letters to systematically break n (in this case 4) separate Caesar ciphers. Once all the separate Caesar ciphers have been cracked, then we have cracked the entire message.

**Cryptography in the real world today**

In the modern world, cryptography is everywhere. Secure transmission over the internet is underpinned by cryptography, sending messages over services such as WhatsApp uses (controversially) end – to end encryption, paying using contactless and many, many other day - to day activities all use high level encryption to keep our information secure in the digital world. In practice, these systems are hugely complex and involved, but in this “hybrid” project we have touched upon (very lightly) how public key encryption and symmetric encryption can be used together to secure data.

In this project, we solely used the RSA encryption scheme. In practice, there are two other families of public key schemes, namely Discrete logarithm schemes and Elliptic curve schemes. Discrete logarithm schemes make use of the discrete logarithm problem in finite fields, and Elliptic curve schemes are a generalisation of the discrete logarithm problem on Elliptic curves, which makes use of point scaling to derive the public and private keys, but we will not venture into that here. In practice, Elliptic curve schemes have efficiency advantages over RSA and discrete logarithm schemes though, as they can provide the same level of security as these other two schemes but with considerably shorter operands, typically 160 – 256-bit ECC keys are comparable to 1024 – 3072-bit RSA/ discrete logarithm keys.

With this project, our symmetric cipher was the Vigenère cipher, but in practice much stronger ciphers are used. By far the most common symmetric cipher in use today is AES, the Advanced Encryption Standard, which replaced an older standard, The Data Encryption Standard (DES). AES uses three possible key lengths, 128 / 192 / 256 bits, all of which are perfectly secure. As well as AES, we can use 3DES (triple DES) which applies the DES algorithm three times over to each data block, and though DES by itself is considered insecure, so far this is resistant to attacks.

This was an incredibly brief overview of cryptography in the real world, but it covers the basics of how we need asymmetric cryptography for key exchange, and symmetric cryptography for the actual bulk data encryption.

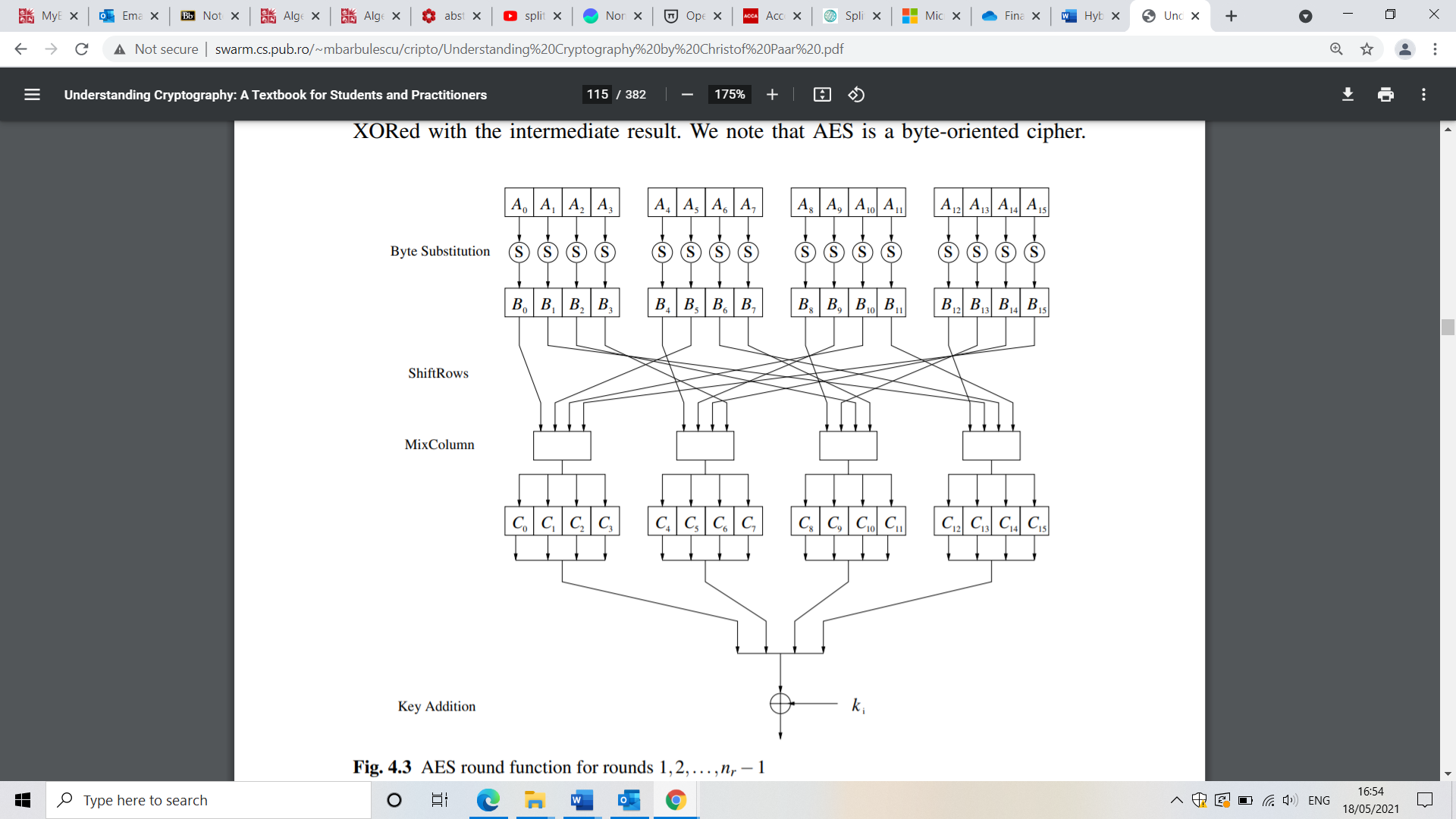
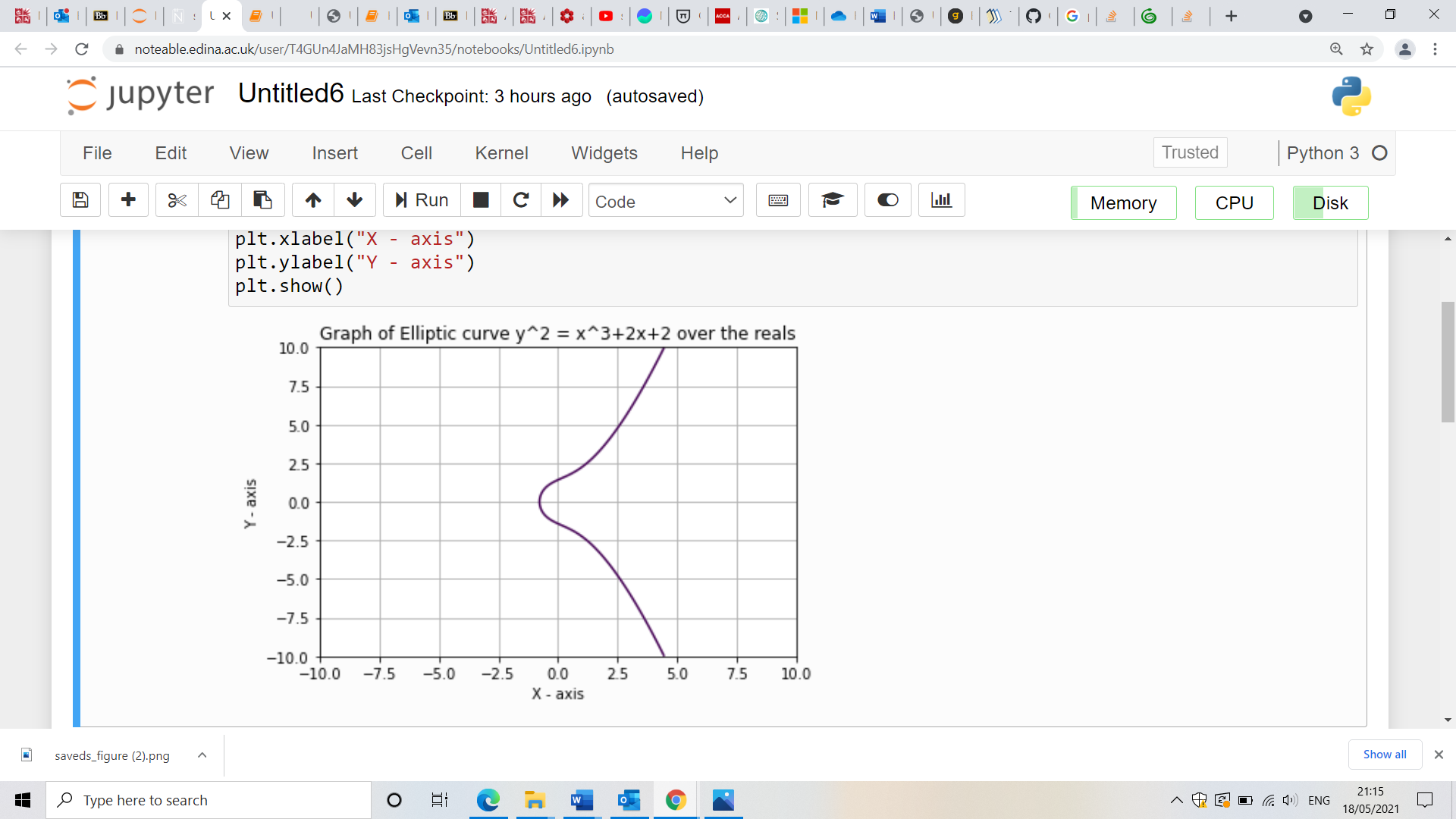


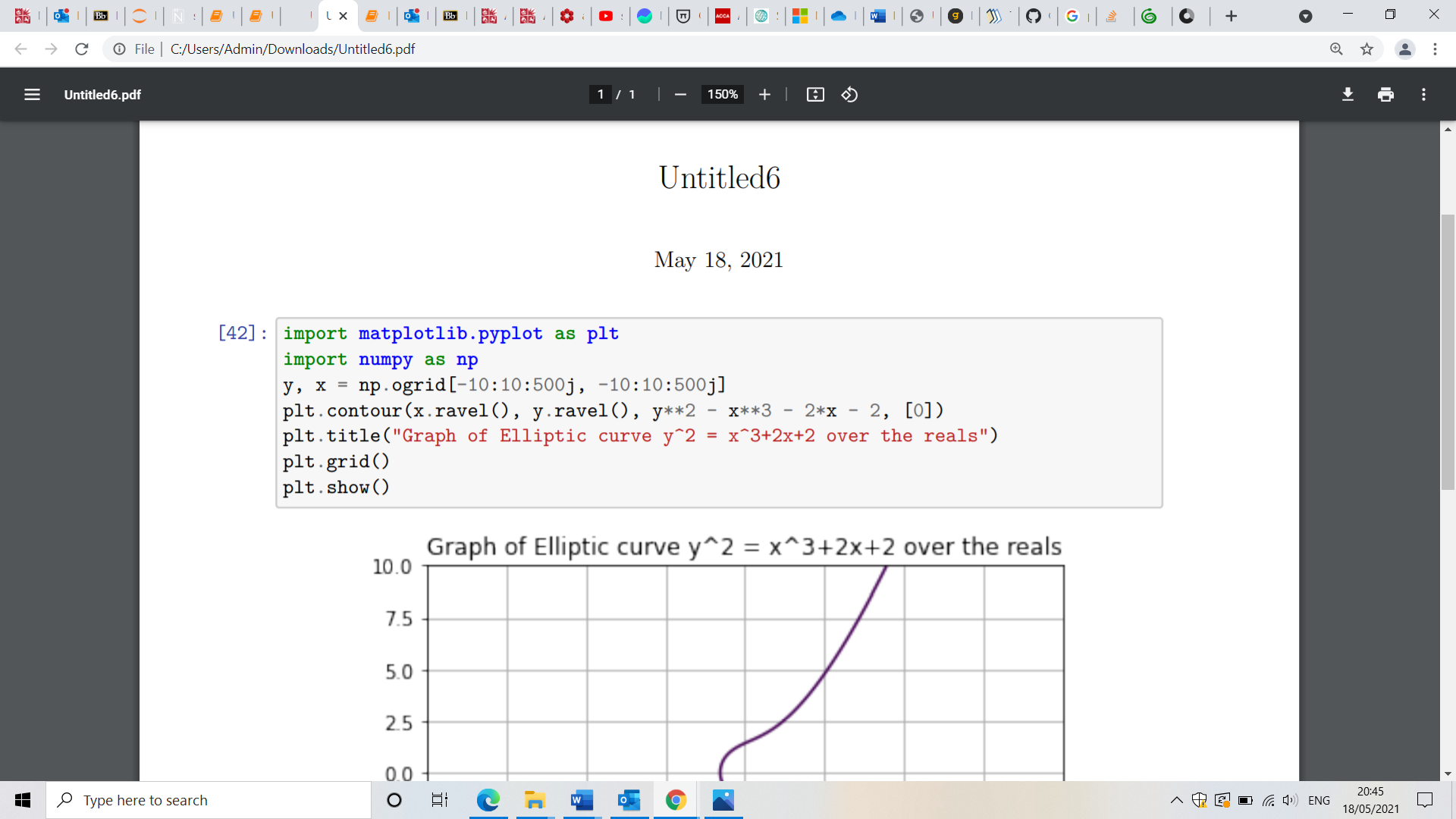
Image: The AES round function, high level overview (Source: Introduction to Cryptography, C.Paar Ruhr University Bochum, Germany)

**Closing note**:

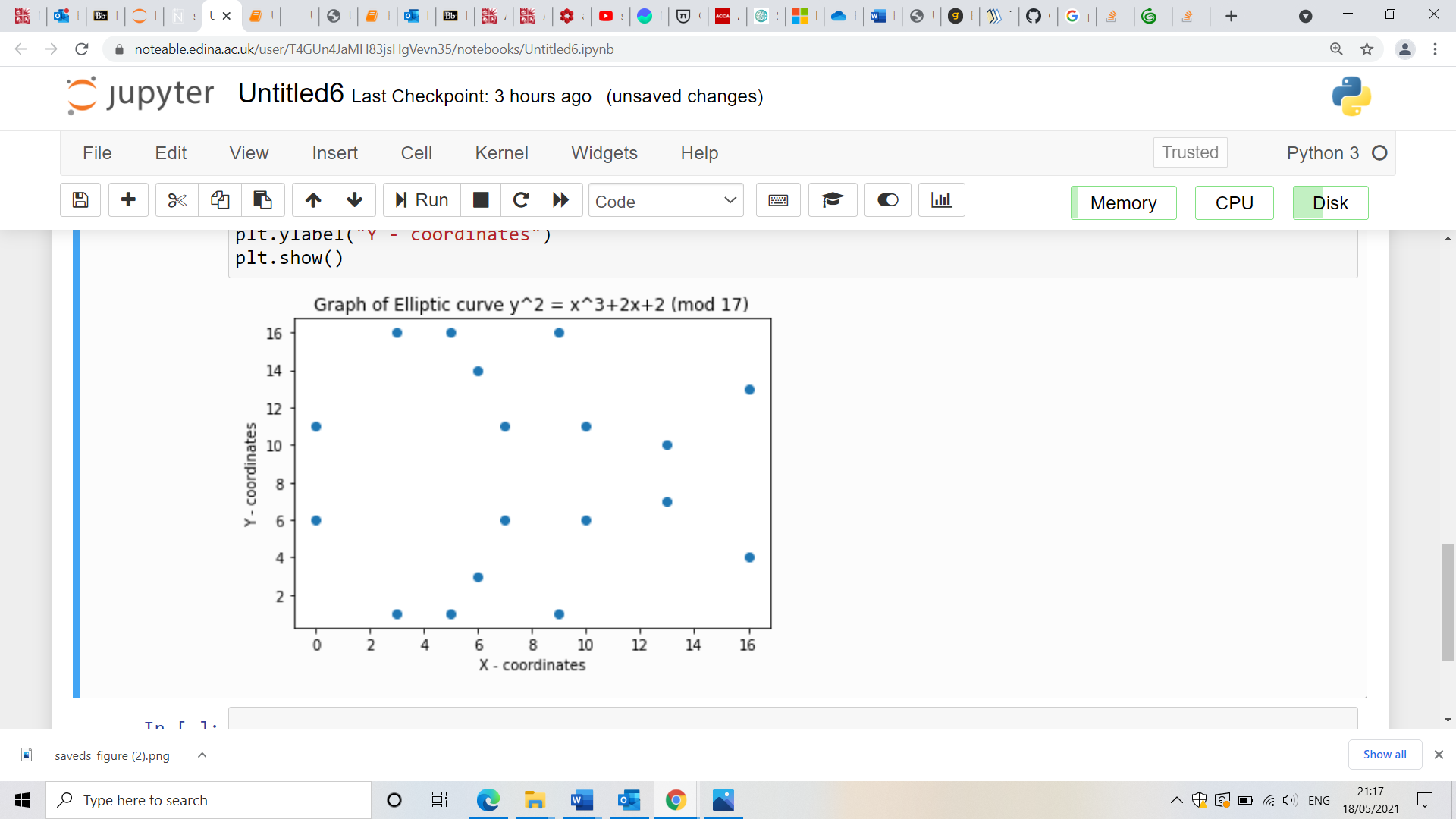
Elliptic curves over finite fields look entirely different to elliptic curves over the reals, as we will show with the following example. Let us consider the curve y^2 = x^3+2x+2. Plotting this curve over the reals, we get the following result:



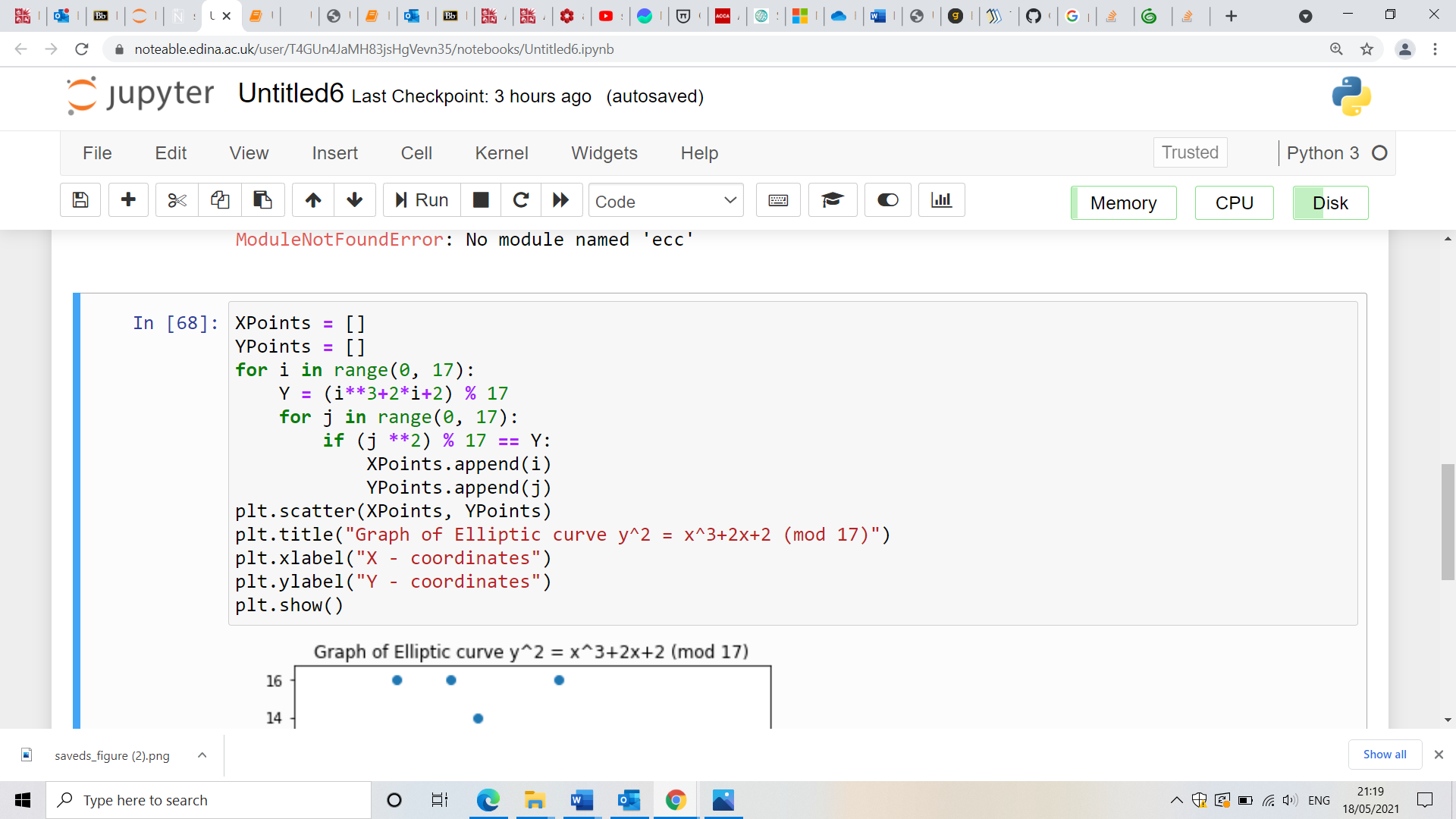
Which I have generated myself in python using the following code:



But, if we now consider the curve (mod 17), so the curve y^2 = x^3 + 2x + 2 (mod 17), we get the following output:



Which has been generated by myself with the following code in python:



**The future of Cryptography**

We will finish with a very brief discussion about the future of cryptography in the real world.

In the not so far future, the field of Cryptography will be transformed by the introduction of Quantum computing. In classical computers today, data is encoded by binary digits that can be 0 or 1 only. Quantum computers work much more efficiently though, they use what are called qubits which can encode data in a superposition of many states. For example, with 3 bits from a classical computer, they can be in one of 8 states at a time, but on a quantum computer using qubits, these qubits can be in all 8 states at once. This has huge implications for the current algorithms we use today to secure our communications.

For example, the key exchanges we use today rely on the assumption of the difficulty of factorising very large integers into the product of two primes, and the discrete logarithm problem. Both problems can be solved relatively easily on a quantum computer, so it is of paramount importance that we develop quantum secure key distribution algorithms to combat this threat. It is interesting to note though there exist certain symmetric algorithms we use today, such as AES – 256, which are believed to be quantum resistant, which is to say that an attack on a quantum computer is not expected to be effective enough to render the algorithm useless if the key sizes are large enough, so even AES – 128 and AES – 192 may be a little insecure in the future.

It is important to stress though that quantum computing is still a bit off yet, but researchers have managed to build a quantum processor that could correctly factor the number 15 as 15 = 3 x 5. It may not seem like much, but it represents the future ahead of us and how once again Cryptography will have to drastically change to enable us to secure communications in the future.

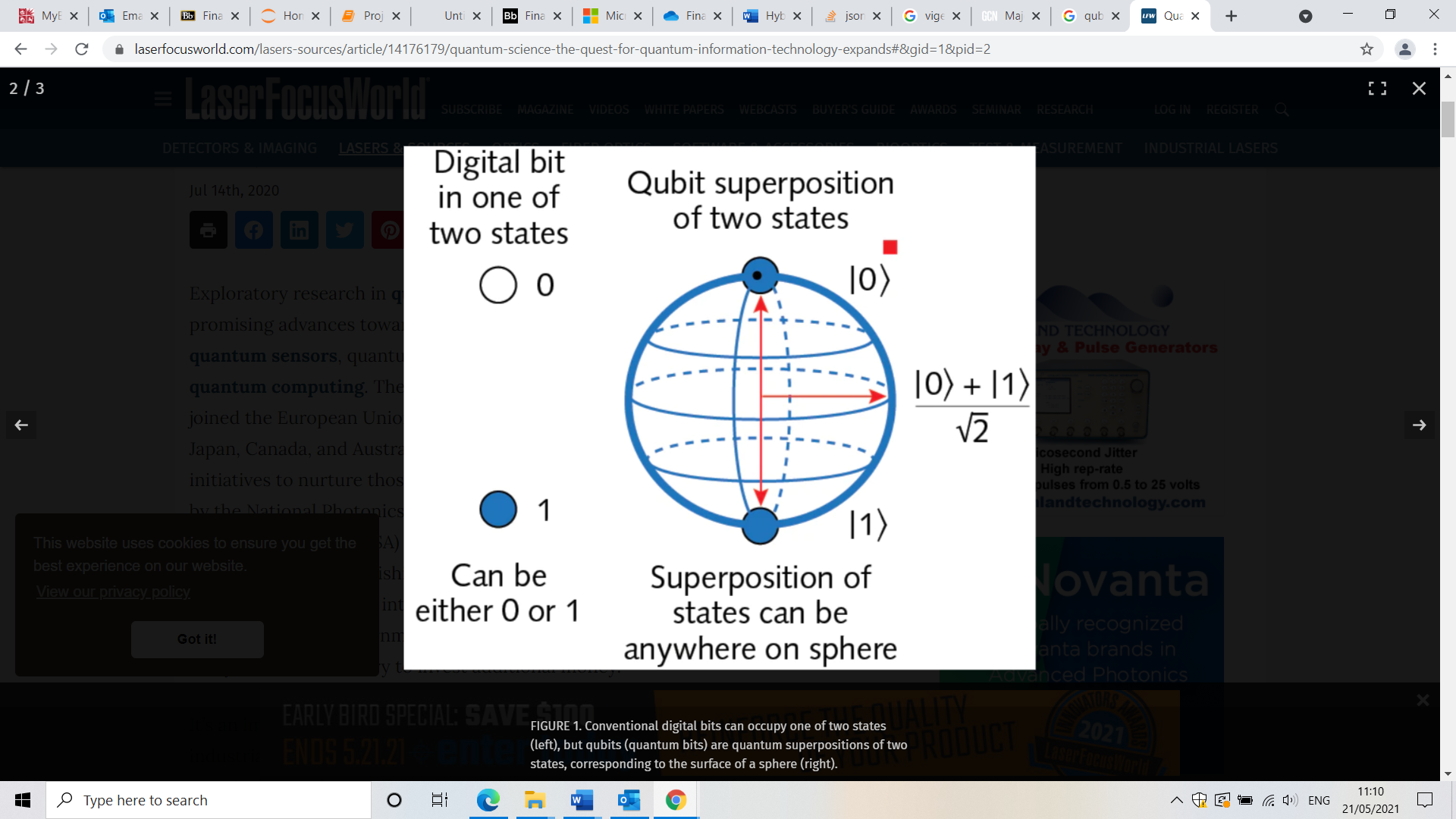


Image: Diagram showing a classical bit vs a qubit. Note how the qubit can be in the superposition of two states, giving it its huge processing advantage over classical bits (Source: www.laserfocusworld.com)