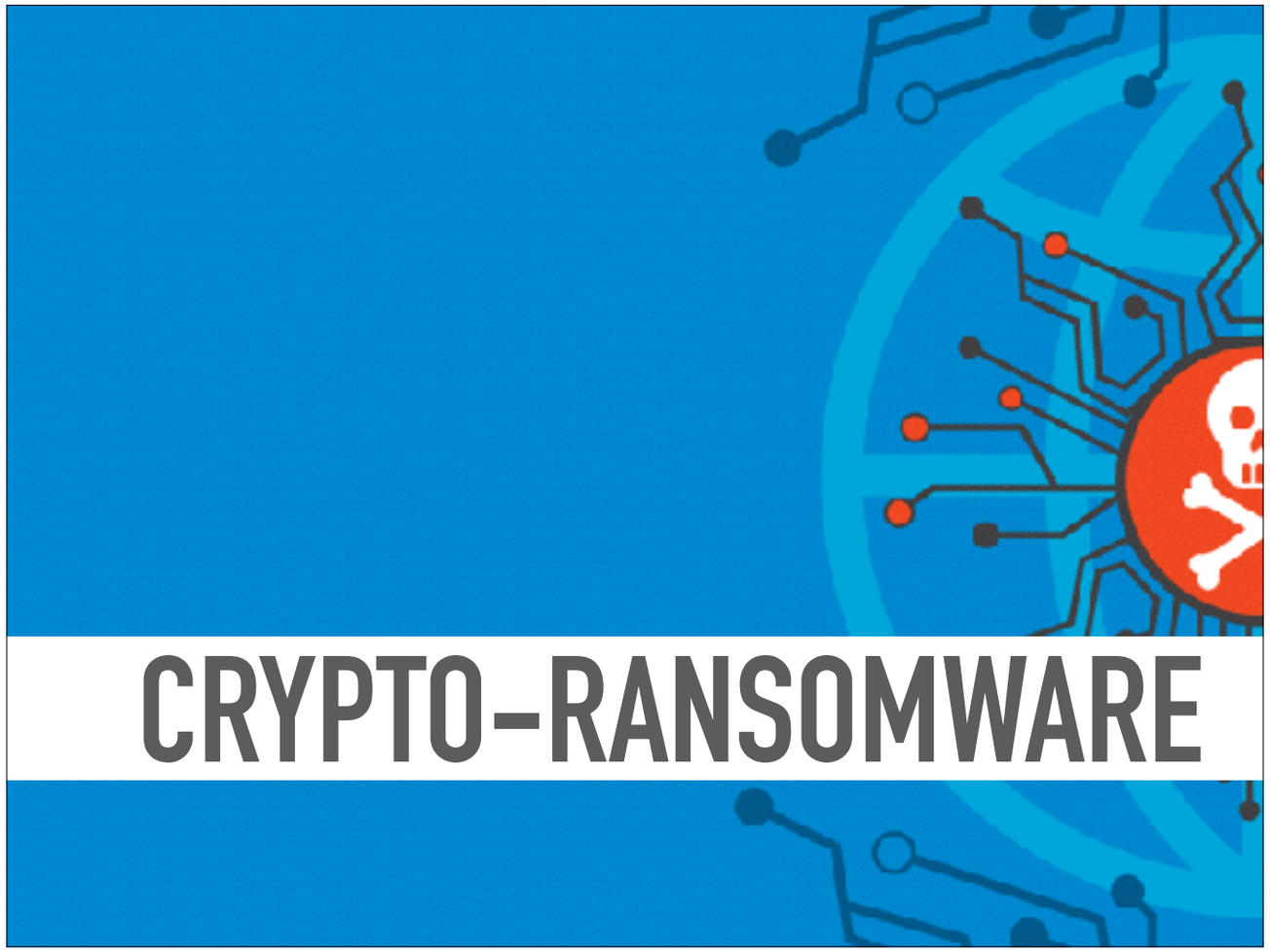
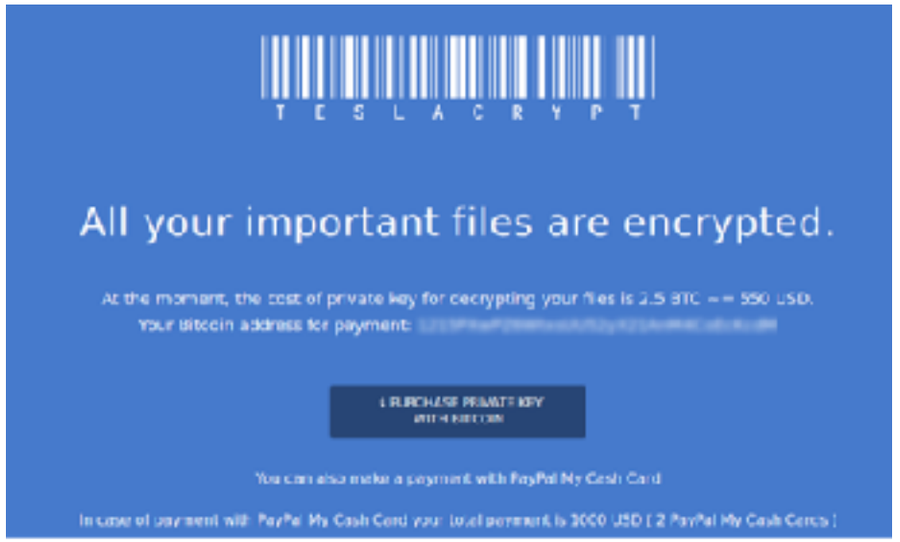

CRYPTO-RANSOMWARE

Crypto-Ransomware

Encryption Algorithms

Algorithms IRL

Today I'm going to tell you about crypto-ransomware and why it’s on the rise, the algorithms that make it possible, and how these attacks are defeated.

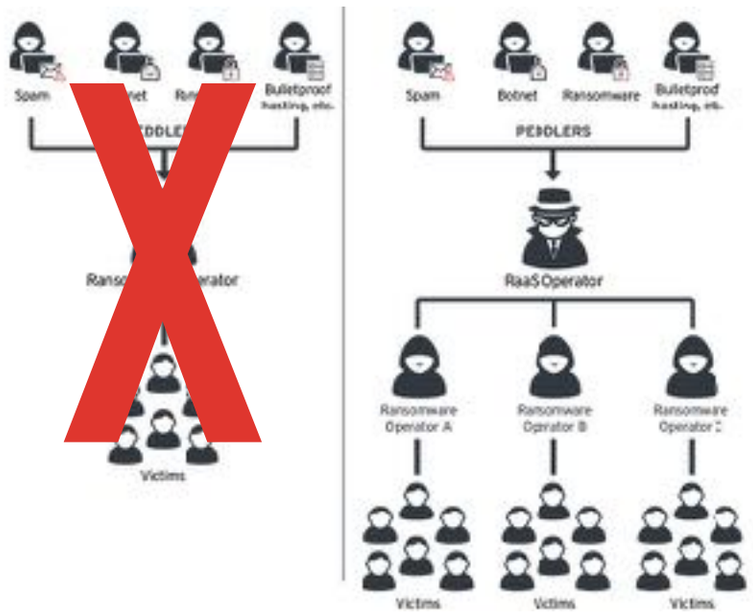


CRYPTO-RANSOMWARE:

Specialized form of malware designed to extort money…

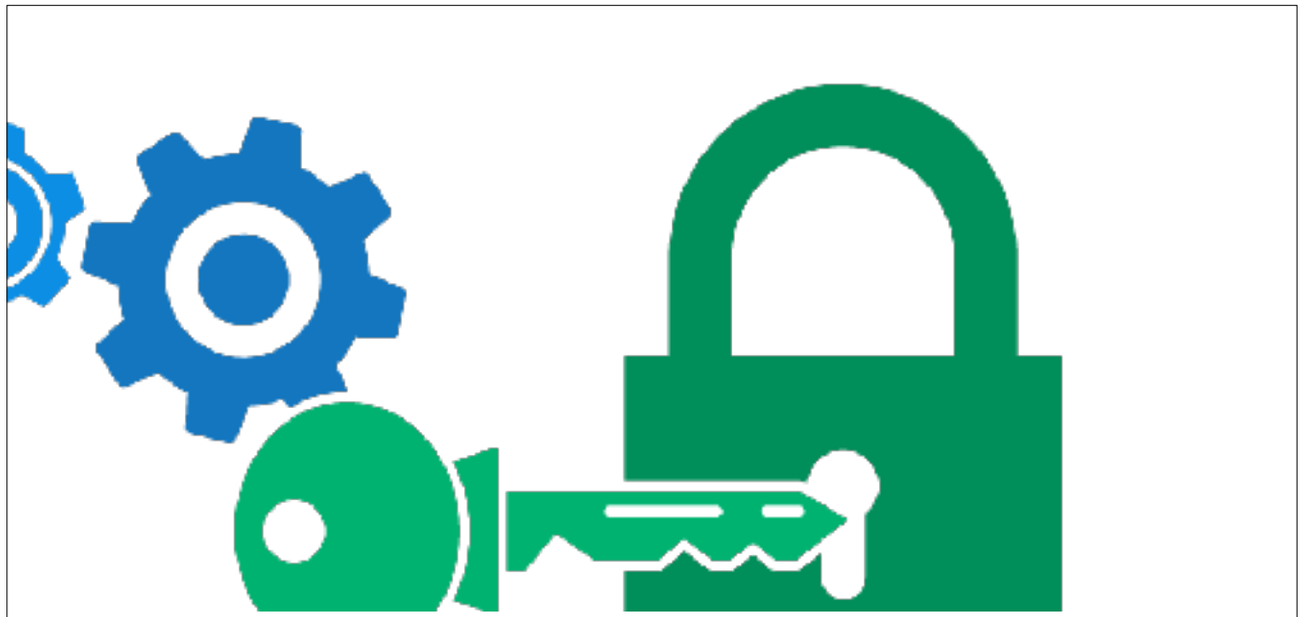
by encrypting files and telling victims they must pay for a key to get the files back

What is ransomware? As the name suggests, it's type of malware that demands money from the victim! Ransomware dominates the malware market and crypto-ransomware, specifically, encrypts your files and doesn't allow you to have your files back until you pay for a key (and then hopefully you actually get your files back).



RaaS: easier than identity theft!

And committing these attacks are only getting easier to do! With the growth of ransomware-as-a-service (RaaS), people no longer need to have the skills to create ransomware or know how to run the operations. They can just download a kit, pay back a % of each profit made to the operator, and they've started a lucrative career as a cyber-criminal.



ENCRYPTION ALGORITHMS

a.k.a ciphers

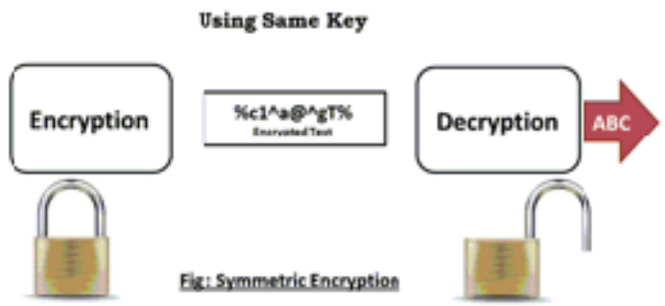
At the heart of all of this is encryption. What makes crypto-ransomware so dangerous is that it uses methods that are designed to be unbreakable…
And we want these methods to be unbreakable because we use these same methods everyday for secure communication, for things like online shopping and text messaging.

Fig : Encryption and Decryption

At a high level, an encryption algorithm, or a cipher, is just a mathematical function that combines an inputted string with another string, called a key, and their combination is the encrypted output.
The goal of any encryption algorithm is to make it as difficult as possible to reverse that output without using the key. So, these algorithms need to create randomness, and then hide the patterns that led to the randomness.

SYMMETRIC ENCRYPTION A.K.A SECRET-KEY CIPHERS

- **One private key** i.e. secret key, for encrypting and decrypting
- Quickly encrypts files, but key can be intercepted
- Example: AES algorithm is the current federal government standard for encrypting classified information

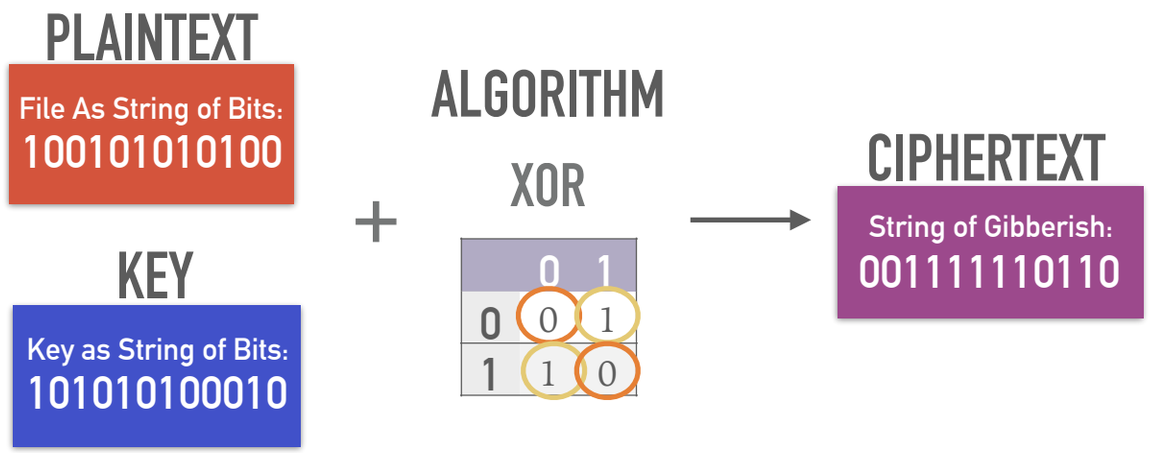


There are two main types of encryption algorithms: the most popular in crypto-ransomware is called Symmetric Encryption, which you'll also hear called secret-key ciphers. It works with one single key that is used to encrypt and decrypt. The functions used are inverses, thus the name Symmetric Encryption.

Cybercriminals like this algorithm because it's the computationally cheapest way to encrypt files, but the price of that cheap processing power is that the key can be intercepted.

BASIC MATH UNDERNEATH SYMMETRIC ALGORITHM:

- **Under the hood:** bit manipulation!
- **XOR:** Either 1 or 0, but not both
  - Modulo 2 addition but numbers are not carried



How does symmetric encryption work? Exactly how is out of the scope of this tech talk, so just keep in mind that symmetric algorithms have a lot more steps than I'm letting on in this slide, but, this is the basic math that make up those steps.

This cipher goes all the way back down to bits, comparing 1s and 0s with "exclusive or," commonly called the XOR operator.

The function compares bit by bit and applies the XOR logic:
- we return 1 if the two bits are different
- if both bits are the same, we return 0

Another way to think about it is that adding the bits together gives the same result as finding the XOR of those values, but the numbers are not carried over.

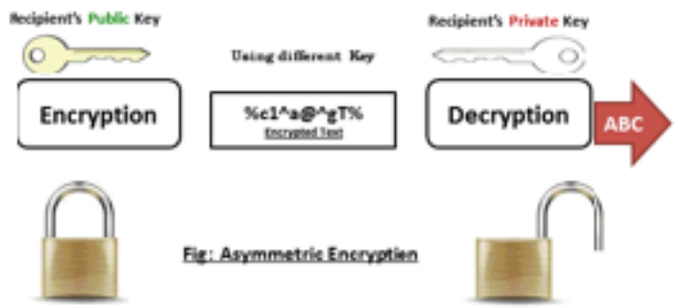
Ee compare file bit to key bit, find the XOR, and the result is our ciphertext!

Why XOR over AND or just plain OR? Well, it makes sure that the amount of 0s and 1s is equally distributed, making it harder to find any patterns. XOR also makes decryption easy - we just XOR the ciphertext bits & the key bits to get the plaintext back.

But what can we do if we don't want the key to be intercepted?

ASYMMETRIC ENCRYPTION A.K.A PUBLIC-KEY CIPHERS

- **Two encryption keys**, mathematically related:
  - **Public key:** stored on the victim's system and encrypts files
  - **Private key:** decrypts files and is stored on criminal server
- More secure but much slower than symmetric encryption
- Example: RSA algorithm is used by most eCommerce sites



Well, that's where the other type of algorithm, asymmetric encryption or public key ciphers, come into the picture. we create security by having two keys instead of one!

One is the public key that's used to encrypt & can be shared, and the other is a corresponding private key that is kept elsewhere.

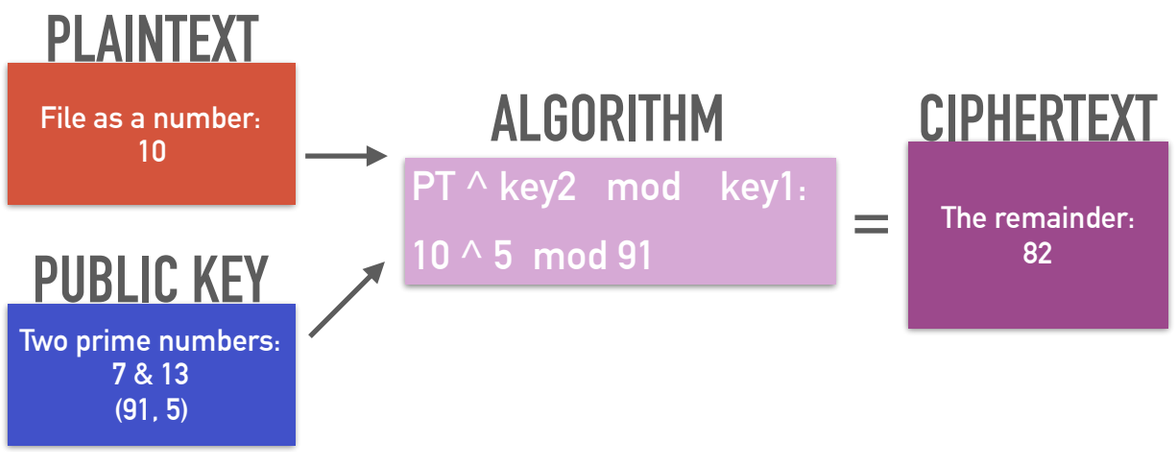
But this type of algorithm has two major drawbacks:

- First is that the two keys need to be a lot larger than those in symmetric encryption for the same amount of security
- This algorithm is much slower: 100-1000x slower than symmetric!

These keys are linked together through math. There's different ways this is done…

BASIC MATH UNDERNEATH RSA ASYMMETRIC ALGORITHM:

- **Under the hood:** two large prime numbers! For encryption:
  - **First value:** Product of X * Y
  - **Second value:** 1 < odd number < common factor of (X - 1)(Y - 1)
- Algorithm function: **modulo addition**!



One way is by using two huge prime numbers - which is how the RSA algorithm, creates its keys. I'm only going showing the process for creating the public key in the slide, but the private key also uses these numbers.

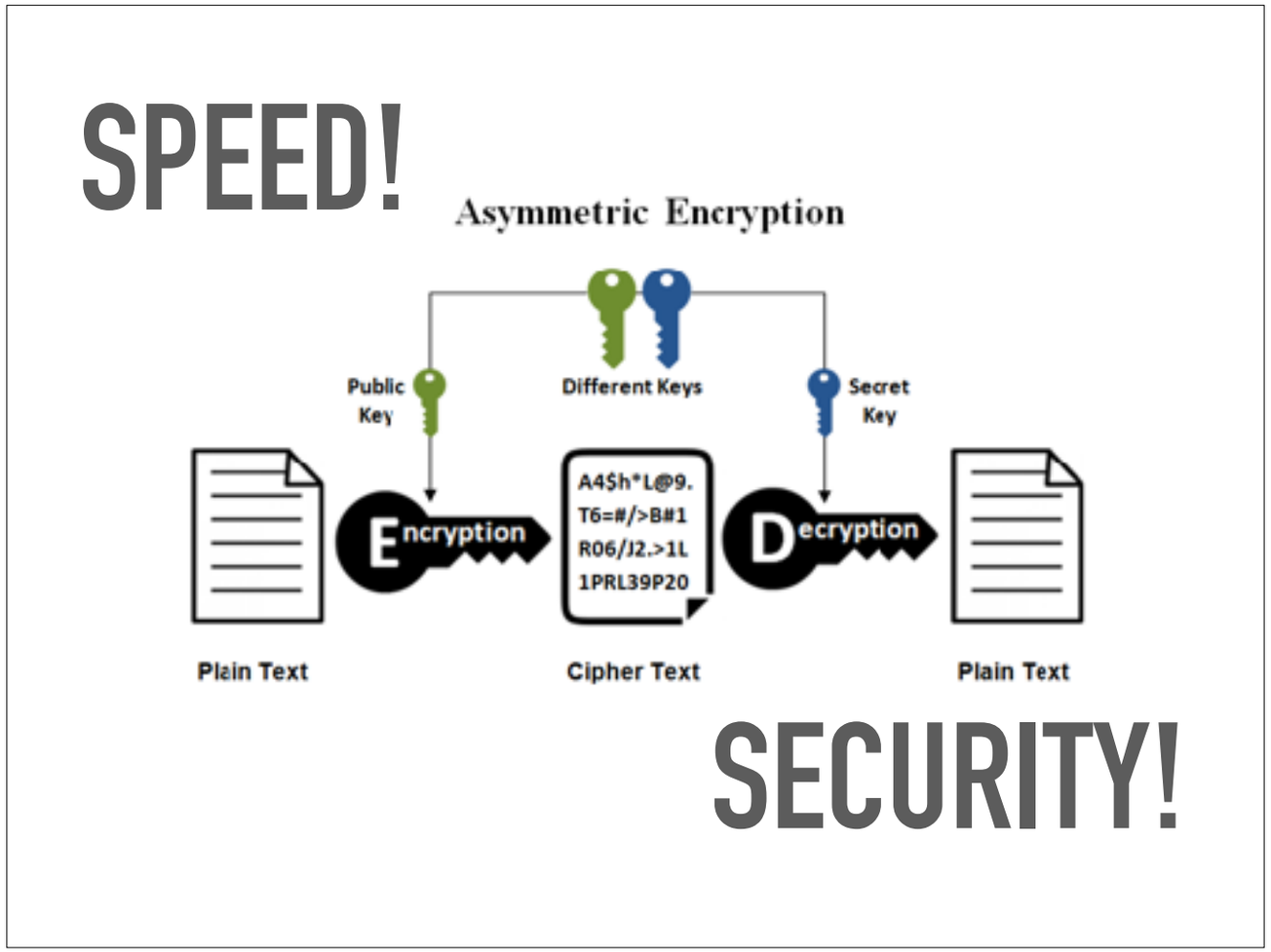
So each key has two values. The first value of the public key is the product of multiplying the two prime numbers. For a secure algorithm, the first value is at least 1024 bits, so at least 309 decimal digits long to give you an idea.

To keep it simple, I'm using 7 & 13, to give us our public key's first value of 91.

The second value of the public key is an odd number greater than 1 that is also mathematically related to the prime numbers - you can see that relationship on the slide. Here I've chosen 5.

The file is also converted to some number and we combine these two by taking the file number to the power of the second value, so 10 to the 5th, and then run that through modulo addition similar to symmetric encryption. But instead of 2 as our modulus, we're using the first value of the public key, 91. The remainder is our cipher text!

The power of RSA comes from that fact that it's easy to multiply two prime numbers together but factoring to try and recover those two numbers is really hard.



Both these algorithms, when implemented correctly, create random-looking outputs, so what's a cyber criminal to do when they want the speed of symmetric encryption but the security of asymmetric encryption?

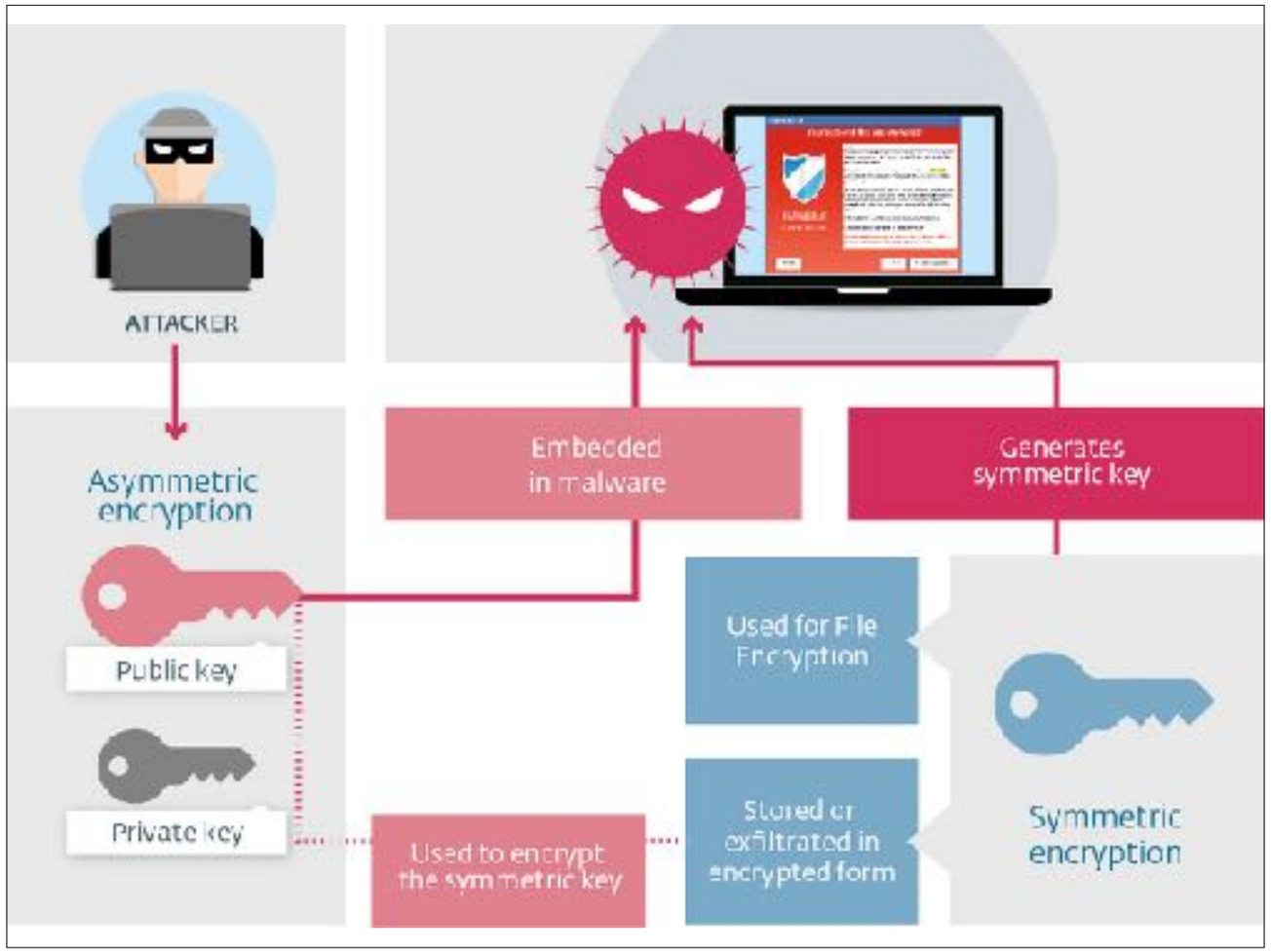
They use both!

BEST OF BOTH WORLDS: MULTI-LEVEL ENCRYPTION

1. Attackers encrypt the victim's files rapidly using a secret key (**symmetric**), offline!
2. Use **asymmetric encryption** to encrypt the secret key

➤ The more secure but slower asymmetric method is needed to encrypt only one file & also secures communication between client (victim) and server

Most ransomware today has some kind of multi-level encryption, using symmetric encryption on the files for the speed and offline access. And then using asymmetric encryption on that secret key and any communication to its server.



This sounds great for the cybercriminals - they have fast performance and tight security, but of course, there are drawbacks.

If the attacker's server can't be reached, then the encryption process can be undone. They also have to make sure that they use a different set of asymmetric keys for each infection,
or else if a victim shares the private key and each victim has the same public key, everyone can get their files back!

CRACKING THE UNCRACKABLE

"The most serious security holes are human, not algorithmic"

So, you may have noticed already the main reason why ransomware fails - human error!
[quote from *The Algorithm Design Manual*]

THWARTING RANSOMWARE BY EXPLOITING MISTAKES:

- **Encryption protocol & Server vulnerabilities**
  - CryptoLocker (2014)
  - *CryptoDefense (v1 2014)*
  - Cerber (v1 2016)
- **Local Keys & Shadow Files**
  - Autolocky
  - *CryptoDefense (v2 2014)*
  - CryptoWall
  - Petya (2016)

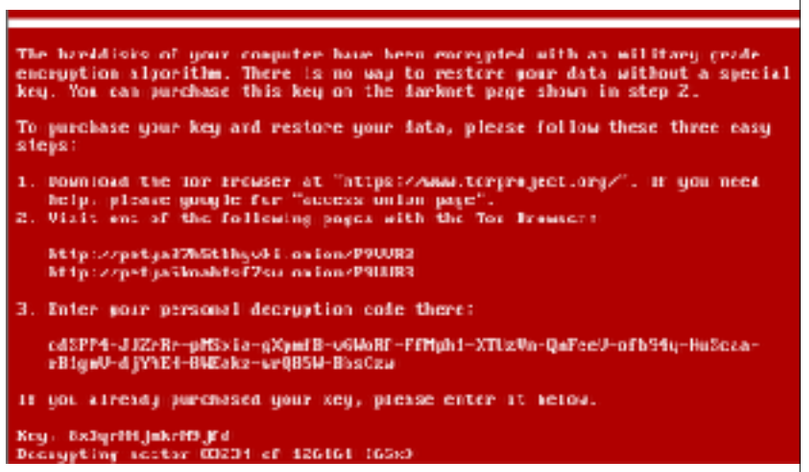
In general, it's easier to hack into a system than to crack a large secret key. So, a lot of the time, the key is uncovered by intercepting it. For example, if we can intercept the ransomware's request to the server, we can get all the information required to recover the key! Which is how the first version of CyptoDefense was cracked.

You'll also see ransomware using victim's built in encryption programs to create the secret keys. It's clever, but sometimes the ransomware doesn't have programming to delete the key or the user's backup files. So, not so clever. CryptoDefense v2, trying to patch up the mistake in v1, did this, so the secret key could be found and was just fed back into the built-in program to decrypt.

So, CryptoDefense, made the same mistake twice - using encryption but giving away the key.

THWARTING RANSOMWARE BY EXPLOITING MISTAKES:

- **Algorithmic errors:**
  - *Petya (2016)*
  - DMALocker (2016)
  - TorrentLocker
- **Bugs**
  - WannaCry (2017)
- **Releasing Keys**
  - BTCWare (5/2017)
  - TeslaCrypt (2016)
  - *Crysis*



I was also surprised by the amount of implementation mistakes that are made. I think it's because of a catch-22 with the keys for both algorithms: the larger the key in the crypto-ransomware, the more secure it is from an attack, but in doing that, there's more places in the program for coding mistakes that help with pattern-breaking.

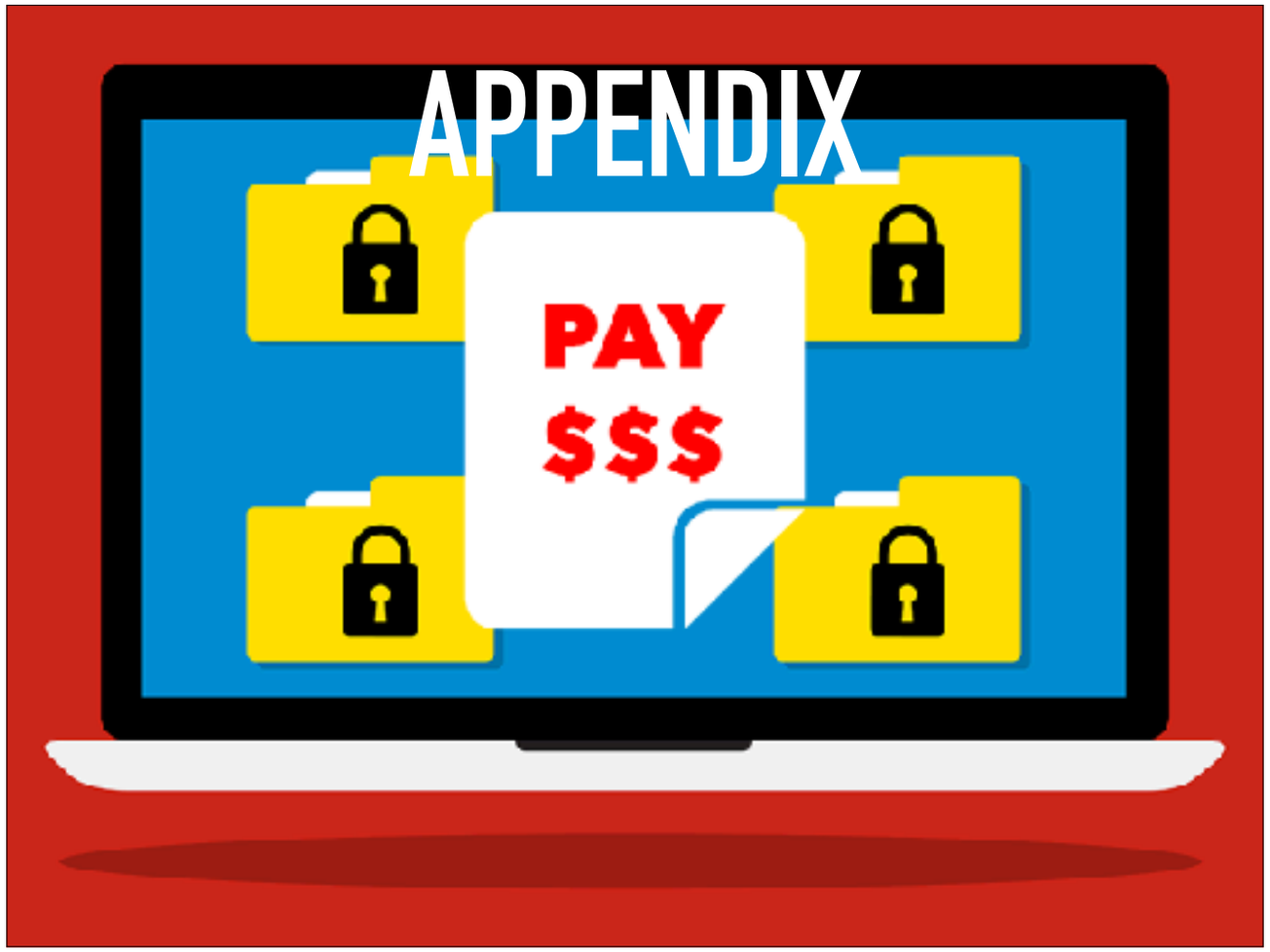
For example, Petya refactored the symmetric algorithm they used in their code, which led to A LOT of mistakes. One of the many was that the 512-bit key they made, ended up having 256 bits of constant and predictable values. Since you can try every possible key combination through a brute force attack - with half the key figured it, that made it much easier to crack.

People will also anonymously release keys online, and some ransomwares, like Crysis, release keys anytime a new version is made as part of their "business" model.

Even though coders of every type make mistakes - like any “good” software vendor, flaws in ransomware get patched up and new versions are released , so be safe and keep your operating system and software up-to-date.

Thank you!

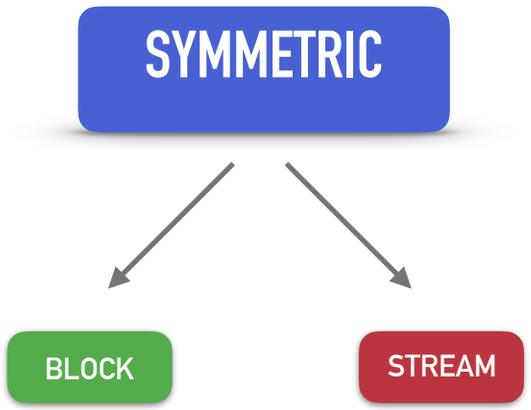
APPENDIX



SECRET KEY: BLOCKS AND STREAMS

More information on how symmetric algorithms work. There are two sub-categories: block and stream ciphers.

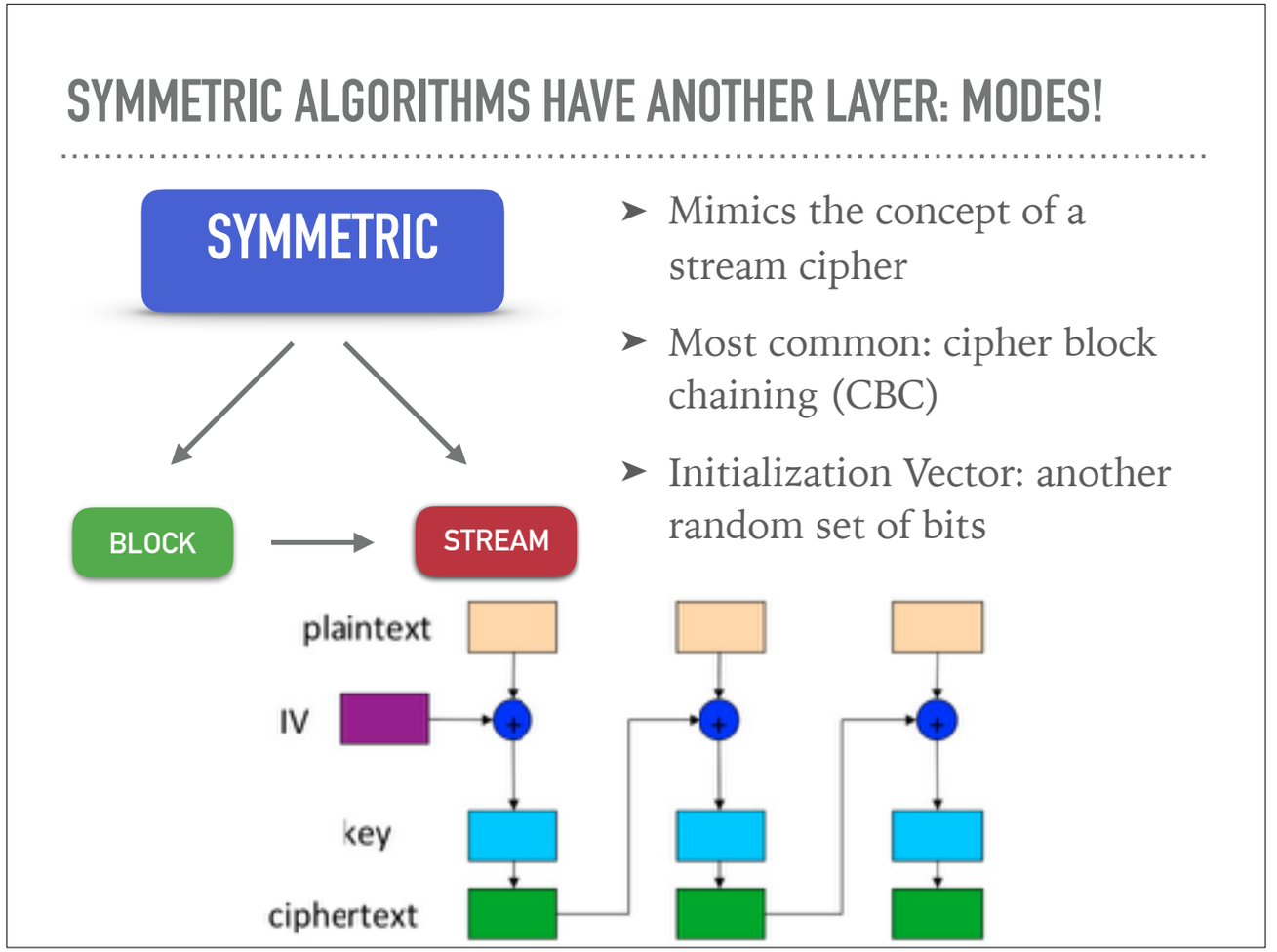
TWO MAIN CLASSES FOR SYMMETRIC ALGORITHMS:



- Stream ciphers:
  - encrypt plaintext as a sequence of bits, one at a time
  - Example: RC4 also used on e-commerce sites
- Block ciphers:
  - encrypt plaintext as a series of blocks, based on fixed byte size
  - Example: AES, the current federal government standard, is a block cipher

Stream Ciphers— encrypt data as a sequence of bits, one bit at a time, thus “continuous streams of plain text”. A stream cipher creates a sequence of bits that doesn't repeat for a very long time, and uses that sequence to hide the message. Downside is an attacker can modify it if he figures out the messages layout. You can do this by systematically making changes to the message's plaintext, switching individual bits on and off in the ciphertext. Block ciphers are more widely used as a result.

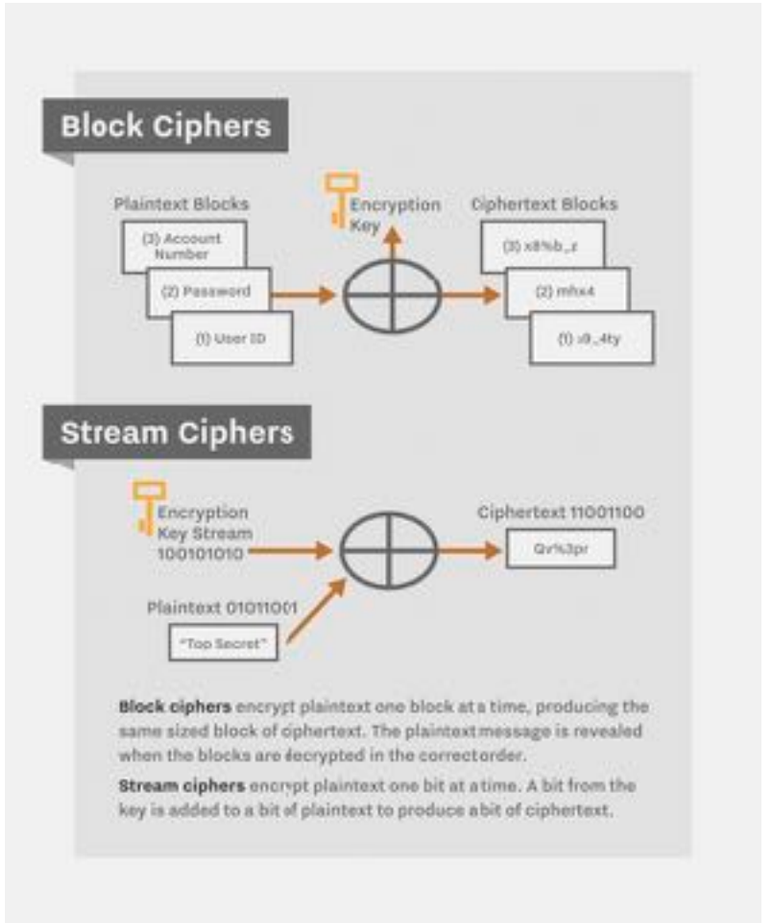
A block cipher encrypts data one fixed-size block at a time -- rather than bit by bit -- producing the same sized block of encrypted data. Unlike stream ciphers, the block cipher spreads a single bit of the plaintext across the entire encrypted block. If an attacker modifies a single bit in the ciphertext, the change will usually ripple through every bit in the block when it is decrypted, creating gibberish. The downside is if we use a block cipher to simply encrypt data one block at a time, then any time we repeat a block of data in the plaintext, we'll produce an identical block of ciphertext. So then you can start guessing what the messages might say.



So, how do we handle these downsides for block and stream implementations? Clearly, another layer of cryptographic manipulation is needed to prevent this problem: Modes!

Block ciphers can be used to build stream ciphers, so you'll often see symmetric algorithms with an appended acronym representing the mode type. An Initialization Vector (IV) - essentially another random string of bits - is used to stretch the key up to the length of the plaintext, becoming what is known as a keystream. Then the keystream is XORed with the message, mimicking the concept of a stream cipher.

CBC, the most common mode, combines the previous block of ciphertext with the next block of plaintext before encrypting it. The IV is applied to the first block before it is encrypted. This chaining mechanism means the encryption of each block depends on the encryption of all previous blocks. CBC helps detect modifications to the ciphertext: any change in the order of the ciphertext blocks will produce a block of garbage when it's decrypted.



A visualization of block and stream ciphers if you’re confused!

AES-CBC 256-BIT CIPHER IN PRACTICE: TESLACRYPT

Only uses one function for encryption, “EncryptFile:”

1. Generates the Initialization Vector for AES, using a GetAndHashOsData API function (which coordinates all key creation)
2. Reads the target file
3. Creates the AES context data structure
4. Encrypts the contents of the file using the algorithm implemented in an “EncryptWithCbcAes” function

Case Study: TeslaCrypt. Breaking down AES-CBC-256:
AES: algorithm name
CBC: mode-type is cipher block chaining
256: how large the key is in bits

“EncryptFile” manages the entire file-encryption process.

At the end, the new encrypted file contains a small header (composed of the AES Initialization Vector in its first 16 bytes followed by the original file size in the next 4 bytes), and then the actual encrypted bytes.

When the process is complete, the new encrypted file is created.

Fun fact: The pop up window displays misleading information: the encryption method is a symmetric AES, and not an asymmetric RSA-2048 as stated by TeslaCrypt in the screenshot above. For asymmetric encryptions, the appended number (here, 2048) actually refers to the *bit size of the modulus* (i.e. how large the product of the two prime numbers is).

RANSOMWARE STAGES

5 STAGES OF RANSOMWARE

1. Deployment
2. Installation
3. Command & Control
4. Destruction
5. Extortion

Great podcast for an in-depth explanation, and the inspiration for this tech talk: https://softwareengineeringdaily.com/2017/04/27/ransomware-with-tim-gallo-and-allan-liska/

STAGE ONE: DEPLOYMENT

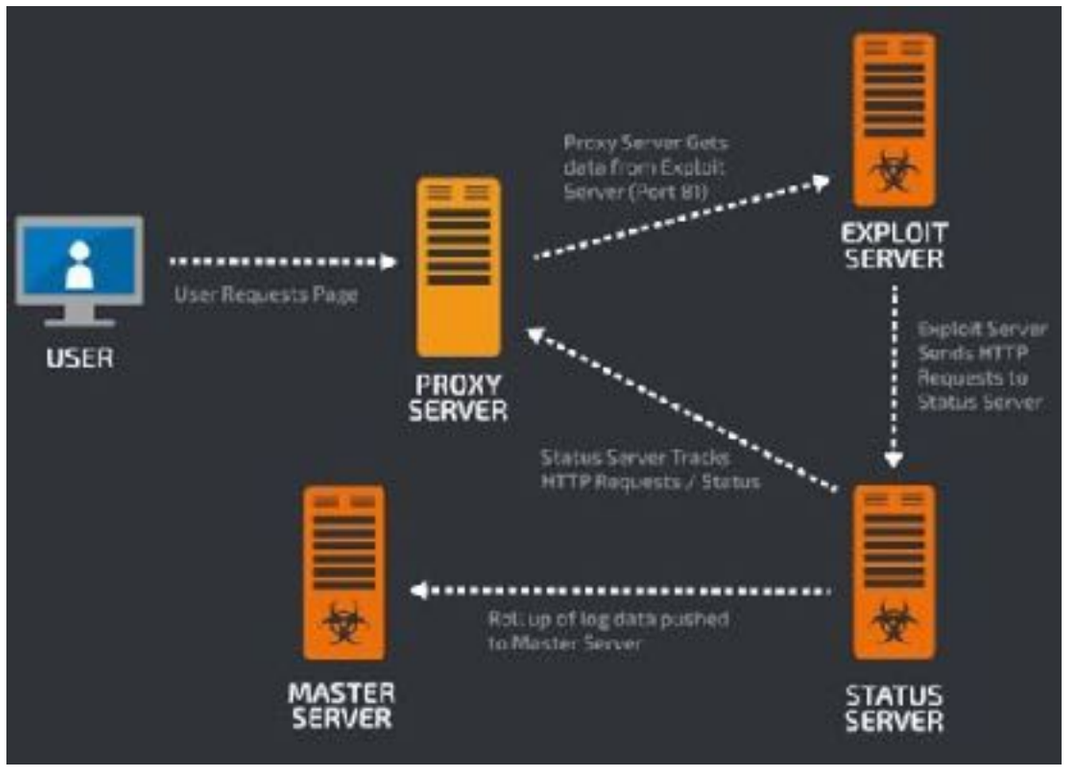
- Two main methods
  - Email:
    - Opening a malicious attachment
    - Clicking a link that points to a webkit
  - Exploit Kits
    - exploit vulnerabilities in software in order to install malware - they rely on outdated or unpatched software
    - compromise third-party web servers and inject iframes into the web pages hosted on them. The iframes direct browsers to the exploit kit servers.

The email attachment either directly installs the ransomware, or initiates a second-stage delivery through a downloader (usually a macro), which subsequently downloads and installs the ransomware .

The spam used to distribute ransomware often poses as an important email from a well-known organization or person, such as:
A notification from the post office or another shipment company, informing the recipient of a delivery
A message from a utility provider about an overdue bill
An alert about the your tax return
Invoices for goods and services
Fake credit card reward schemes
A message from a high-level company executive

Attackers can also redirect users to EKs in the following ways:
Malicious links in spam email, social media posts, SMS
Malvertisements
Redirected web traffic from traffic distribution services

DEPLOYMENT CHAIN EXAMPLE:



STAGE TWO & THREE: INSTALLATION, COMMAND & CONTROL

- Each type of ransomware "family" has its own methodology once it's gotten into a system. Very generally and untechnically, ransomware will:
- Connect to server and ensure that it's got the appropriate code and permissions as well as ensure the communication paths are clear. Then it will begin the process of identifying the files that it wants to encrypt
- Some of the ransomwares wont contact a command and control to avoid getting caught ('disrupted') by someone defending the network
- Finally, as all your files are being encrypted, the ransomware also calls home and transmits a copy of the serial key to one of their command-and-control servers.

Normally, ransomware contacts its command and control (C&C) server, which generates an RSA key pair and sends the public key back for the malware to use in the encryption stage.

FUN FACT: Ransomware can exploit specific vulnerabilities inside the browser that you're leveraging and use that to sort of choose which ammunition that they're going to use to drop a piece of ransomware (like location). computer is compromised, Cryptowall reports back to a command-and-control (C&C) server with the IP address of the infection. The server performs a lookup of the IP address and determines the country that the infected computer is located in. Then, based on various factors, the price returned to the infected computer is adjusted to suit the location.

In the case of file encryption, primarily, they're looking for things; Words documents, looking for JPEG, bitmap files, looking for things that are of value, Office documents, stuff like that. It does blanket searches across the environment for those files, begin to identify them and then starts the encryption process.

Generally speaking, what happens is once the ransomware is installed and everything is encrypted or even during the encryption process, the ransomware may have to reach out to the command and control server to get the private key in order to do the encryption. Once it's done, it sends a note over to the command and control host saying, "Hey, we've infected this system," and it gives the details about the system that's infected. The two big reasons to check in:

-METRICS: so that the ransomware developers can keep track of the systems that are being infected, how it got infected, what the history of the installation work was like, what worked, what didn't work, and so on. the check-in also allows RaaS users to verify that, "Yes, I've had this done." They can go check the portal and see how many victims have successfully had the ransomware installed. At that point, the check-in is simply, "Hey, we're in. Here's the information about the system and the files have been encrypted," or "We weren't able to encrypt everything. I was disrupted."

COMMAND AND CONTROL

- There are 2 main methods that a ransomware will operate: centralized or decentralized.
  - Centralized: communicated with Command and Control server to store all of the keys, handle payments, etc.
  - Decentralized: store the key on your computer, then will charge you for the decryption of that key to then decrypt the rest.
    - More advanced because it allows for offline encryption, but it's also a lot easier to mess up.

Centralized: It will send information to that server when you are infected. If this information can be intercepted, by a corporate firewall, for example, it can yield the key

Decentralized: Decentralized malware will generally encrypt your key with something like RSA and will keep it on your computer. It then tells you to email someone to get your key back. If there really is someone there, they will ask you to send them the encrypted key, they will decrypt it, and they will send you the decrypted key and the decryptor to put it in. If the RSA encryption is messed up, then they won't be able to recover the key and there is no point in paying. If the write is messed up, the same might apply or you might get lucky and find your plaintext key in the mix. Again, it really depends on the author and their implementation.

NON-TARGETED ATTACKS:

1. Deployment:
    1. iframe injection
2. Installation
3. Command & Control
    1. reach out to the servers
    2. encryption algorithms
4. Destruction
5. Extortion

➤ Note: These attacks are pretty automated

From Barracuda: Because of the ease with which non-technical cybercriminals can enter the market, we anticipate a growing trend toward two distinct focus areas for ransomware criminals: a) “low-end” ransomware that demands a few hundred dollars in ransom that is sent by amateurs and other low-level criminals using basic phishing techniques; and b) “high-end” ransomware sent by more sophisticated cybercriminals and focused on high value targets in the healthcare, financial services, insurance and other industries that are more likely to pay significant sums to recover their encrypted data.

TARGETED ATTACKS:

- Servers are an ideal way to target businesses:
  - Brute-forcing credentials for RDP servers
  - Targeting vulnerabilities in web plugins to gain access to web server
  - Exploiting flaws in JBoss servers
- Move laterally across the network to infect numerous computers or find valuable targets such as databases to amplify the impact of an attack, which also allows for reconnaissance
- Use legitimate tools to keep a low profile
- Delete backup files to prevent victims from recovering affected data
- Targeted attacks require a lot of work on the part of the attackers, but this is balanced by a potential for higher profits

The SamSam attackers use freely available tools, such as the opensource testing tool JexBoss, to identify vulnerable servers. Once in, the attackers may steal credentials and conduct further reconnaissance before encrypting any files. The use of open-source and well-known tools can help threats stay under the radar, as many of the tools, such as Microsoft's Sysinternals, which was used by SamSam, are commonplace on enterprise networks.

Symantec case study: To gain entry during the first stage of the attack, the cybercriminals used both watering hole attacks and spear-phishing emails with malicious attachments. The attackers then used back door malware and freely available penetration testing tools to consolidate their position within the network and proceeded to compromise administrator account credentials. They then used the credentials to compromise file, application, and email servers within the company, as well as multiple workstations.

STAGE FOUR & FIVE: DESTRUCTION & EXTORTION

- Destruction refers to one of two things:
    1. Getting rid of any identifying information of how they got into your system.
    2. Then, there's some ransomware that isn't really ransomware, it's just there to destroy files. It will popup a ransom note, but there are no files to recover.
- Extortion: The victim pays the ransom with however many Bitcoin it is. They're given a key. They enter the key and that allows them to decrypt the files, hopefully.

on first point: The identifying information about the ransomware is there ("you've been hacked by Cryptolocker!"), but they don't necessarily want to leave behind how they got on the system.

RANSOMWARE: PLETHORA OF LANGUAGES

- Several new ransomware families have been coded in different programming languages, such as JavaScript, PHP, PowerShell, or Python. Attackers used these languages in an effort to evade detection by security products
  - Powershell and Microsoft Word Macros
  - Python: Zimbra ransomware
  - Javascript & Node.js:
    - Ransom32: up-and-coming RaaS
    - RAA: is currently being distributed via emails as attachments

Ransomware is traditionally written in Assembly / C++, but that has changed.

Powershell: installation of the ransomware begins after someone opens a booby-trapped Word document, which runs macros to download and run the software

Zimbra: targets the Zimbra email message store folder and encrypts all of the files located within it. Most likely installed via the developer hacking into the Zimbra server and executing the Python script. Once the script is executed it will generate a RSA key and a AES key that is unique to the victim's computer.

Ransom32: developers of Ransom32 take a 25% cut of all ransom payments and then forward the rest to the bitcoin address an affiliate entered when they joined the affiliate program.

RAA: When the JS file is opened it will encrypt the computer and then demand a ransom of ~$250 USD to get the files back. To make matters worse, it will also extract the embedded password stealing malware called Pony from the JS file and install it onto the onto the victim's computer.

DEALING WITH ATTACKS: INFORMATION ARCHITECTURE

- "What are the impacts to business operations if we are attacked?"
  - Factors to take into account:
    - "How do I respond to ransomware?"
    - "Do I have effective backups?"
    - "Am I doing sort of continuous backup and monitoring of the devices?" "What's the size and scope of the infection?"
    - "Do I have map to drives all over my environment, and because this system was infected, it's actually infected a number of my servers as well because they've had map access to these drives?"
  - Above approach makes it possible to recover files from a backup, but one of the things that happens a lot though is people don't test their backups.
    - If you're using tape, are the tapes available, or are they sitting there somewhere else? If you're a disc, have those discs been compromised? Do you know if they have or haven't?
    - Has your detection of the ransonware map to when the last backup was? Did you actually backup the locked files unintentionally?

A preventative approach for businesses in setting up architecture

ADDITIONAL RESOURCES

RANSOMWARE SPECIFIC RESOURCES

- **Digital Guardian's "A History of Ransomware Attacks:"** https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time#4
- **Barracuda Whitepaper:** https://assets.barracuda.com/assets/docs/dms/Best_Practices_for_Dealing_With_Phishing_and_Ransomware_-_Barracuda.pdf
- **Symantec Whitepapers:**
  - https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf
  - http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf
- **How Ransomware Infects a Computer:** http://911ransomware.com/2017/03/14/ransomware-infect-computer/
- **Ransomware Anatomy:** http://911ransomware.com/2017/03/06/anatomy-of-a-ransomware/
- **Exploit Database:** https://www.exploit-db.com

ALGORITHM SPECIFIC RESOURCES

- **Understanding the basics:** http://searchsecurity.techtarget.com/Understanding-encryption-and-cryptography-basics
- **Implementation of AES for Developers:** http://www.adamberent.com/documents/AESbyExample.pdf
- **Math behind RSA Algorithm:** http://www.mathaware.org/mam/06/Kaliski.pdf
- **Intro to Crypto Algorithms:** http://911ransomware.com/2017/01/31/different-types-of-crypto-algorithms/
- **Encryption Fundamentals:**
  - http://www.infosectoday.com/Understanding_Cryptography/Articles/Fundamentals_Cryptography_Encryption.pdf
  - http://www.encryptionanddecryption.com/algorithms/encryption_algorithms.html
- **Cryptography Intro from Northeastern University:** http://www.ccs.neu.edu/home/noubir/Courses/CS4740/F10/slides/cryptography.pdf

RANSOMWARE TECHNICAL BREAKDOWNS

- **Evolution of attacks:** http://resources.infosecinstitute.com/ransomware-in-the-wild-its-an-emergency/
- **Overview of 10 different strains:** http://resources.infosecinstitute.com/a-brief-summary-of-encryption-method-used-in-widespread-ransomware/#gref
- **Petya, Jigsaw and Autolocky:** https://www.welivesecurity.com/2016/04/28/ransomware-is-everywhere-but-even-black-hats-make-mistakes/
- **CryptoWall:** https://www.botconf.eu/wp-content/uploads/2015/12/OK-P14-Yonathan-Klijnsma-The-Story-of-CryptoWall-a-historical-analysis-of-a-large-scale-cryptographic-ransomware-threat.pdf
- **TeslaCrypt:** http://blogs.cisco.com/security/talos/teslacrypt
- **Petya:** http://blog.checkpoint.com/2016/04/11/decrypting-the-petya-ransomware/