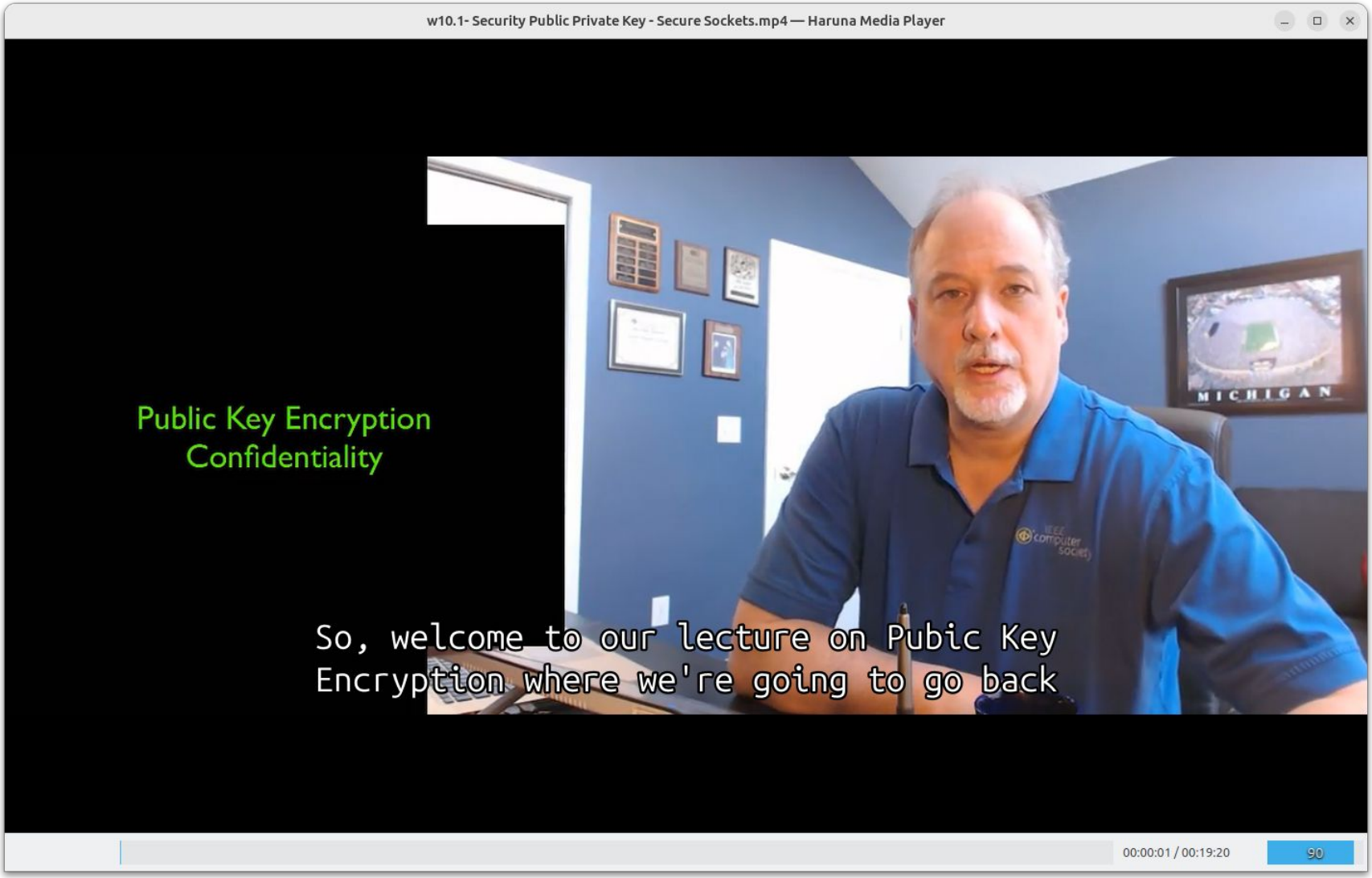
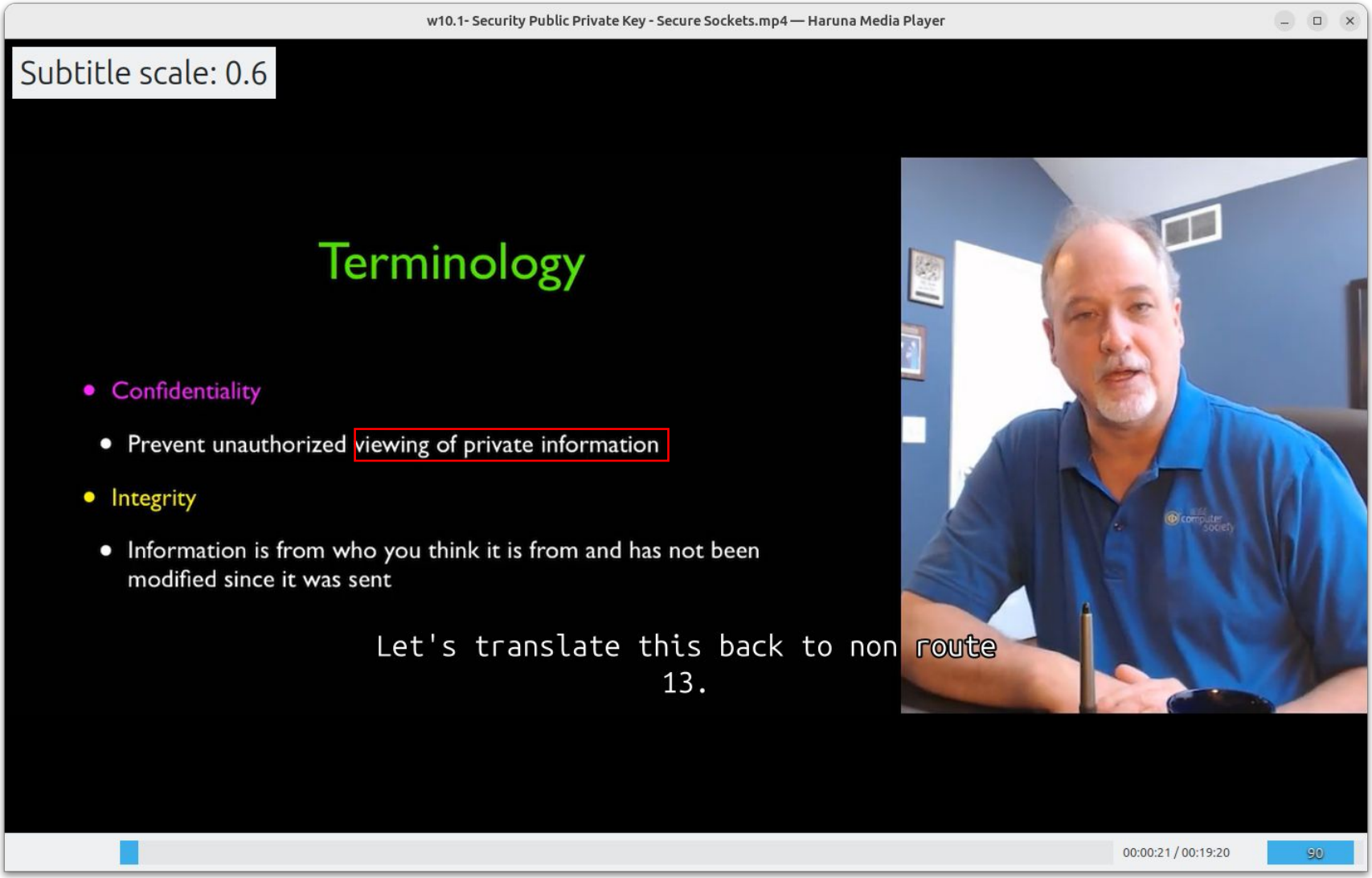
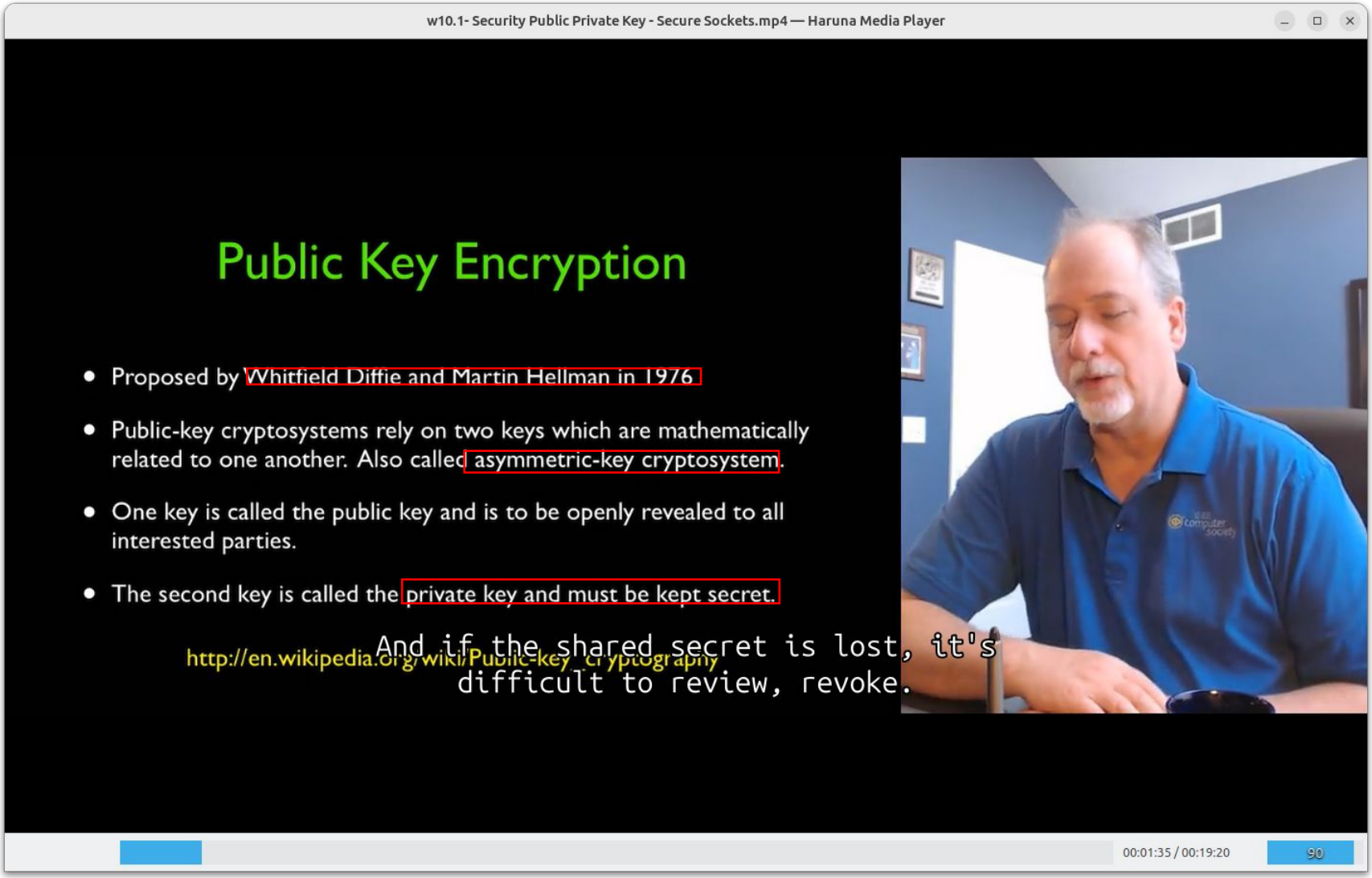
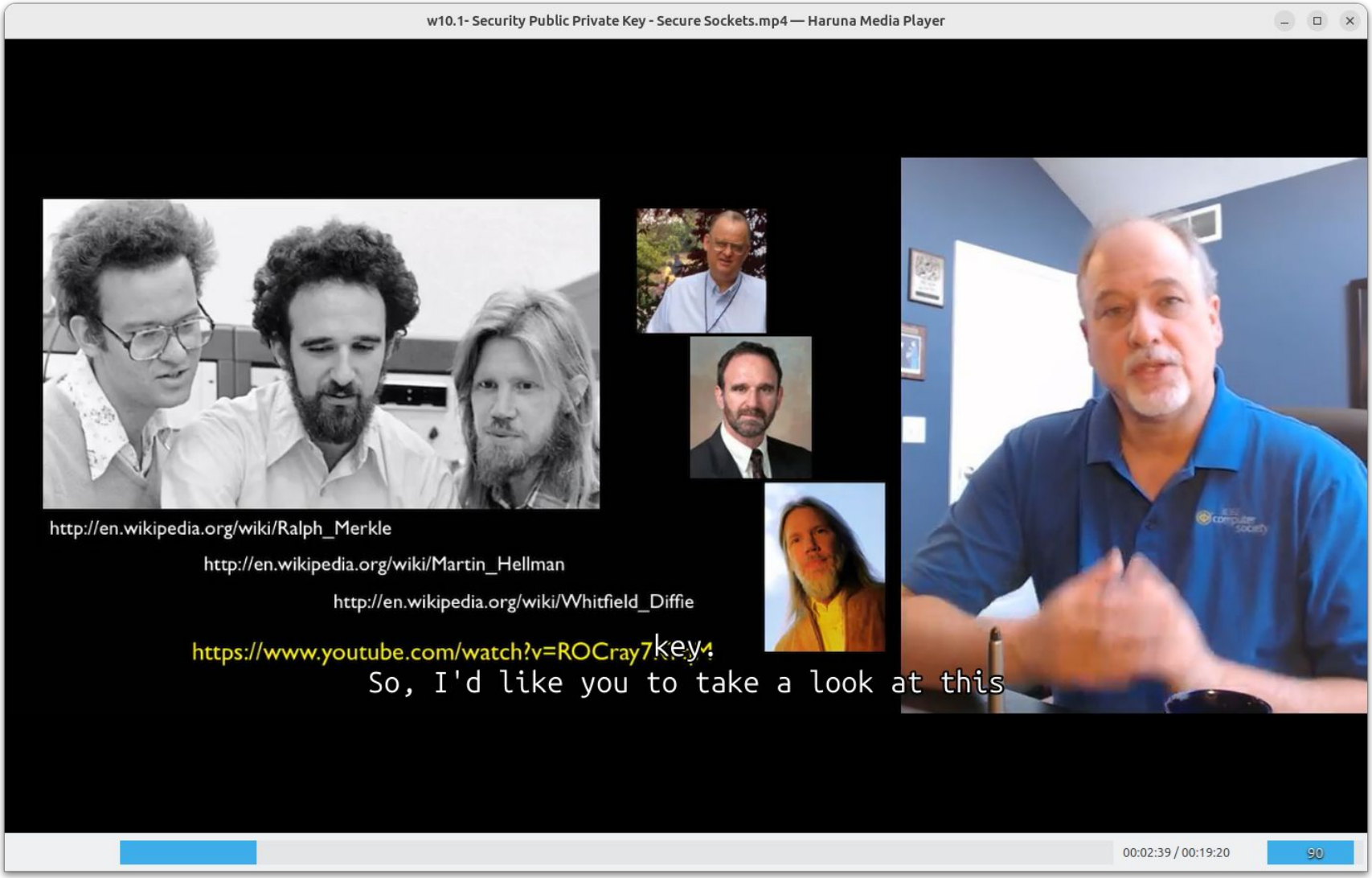
w10.1- Security Public Private Key - Secure Sockets









Public Key

- A message encrypted with one of the keys can only be decrypted with the other key.
- It is computationally infeasible to recover one key from the other
- Public-key cryptosystems solve the problem of secure key distribution because the public key can be openly revealed to anyone without weakening the cryptosystem.

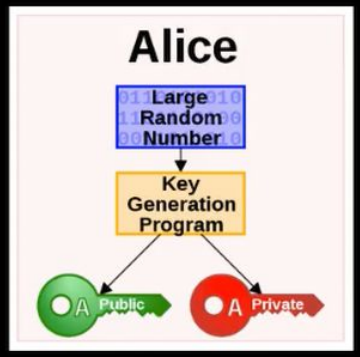
have this public key.
So, the public key is part of a public

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

Subtitle scale: 0.3

Generating Public/Private Pairs

- Choose two large* random prime numbers
- Multiply them
- Compute public and private keys from that very large number



* The definition of "large" keeps getting bigger as computers get faster

key encryption, you have to generate a pair.

00:05:19 / 00:19:20

90

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

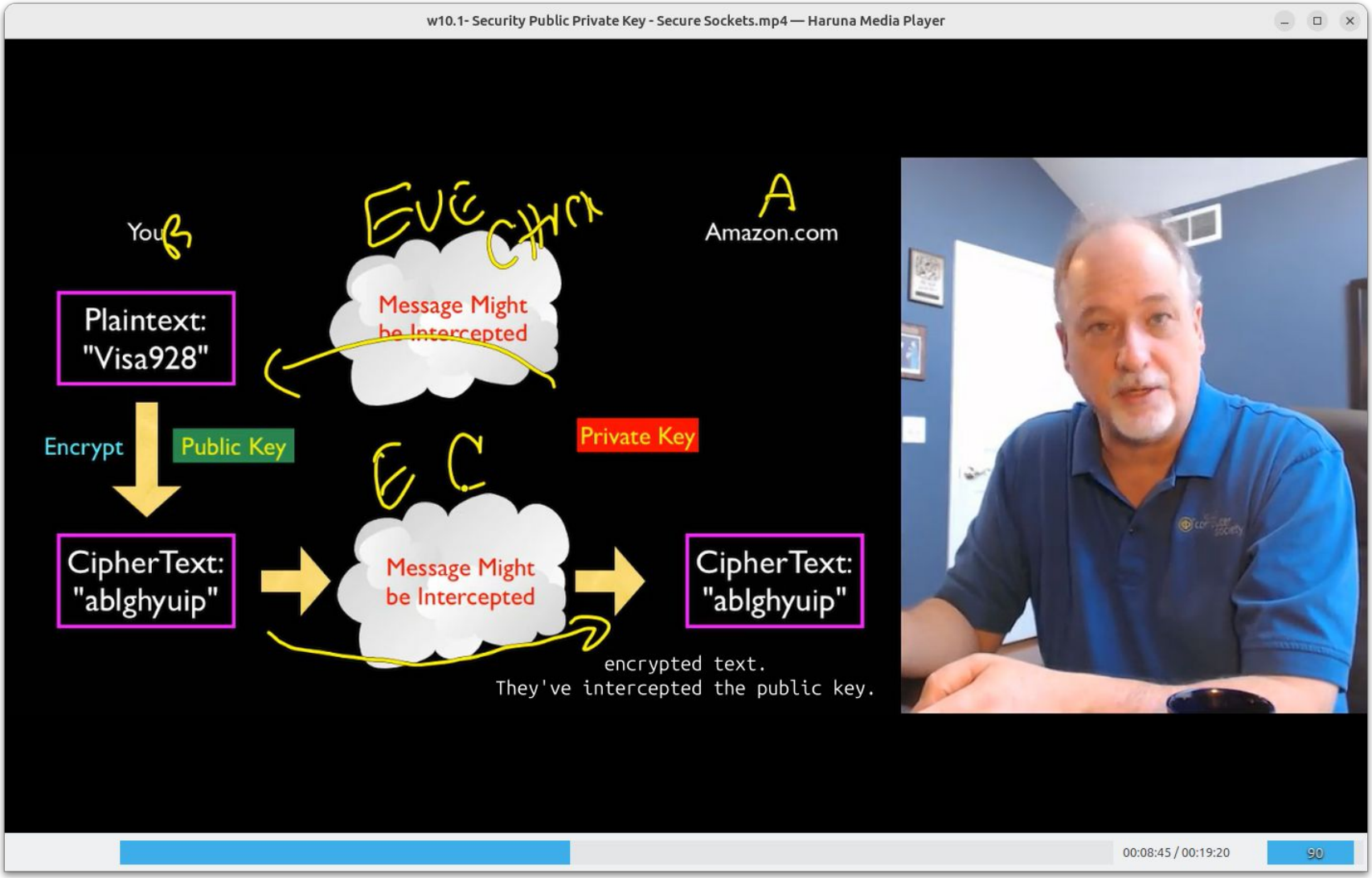
Public Key Math (light)

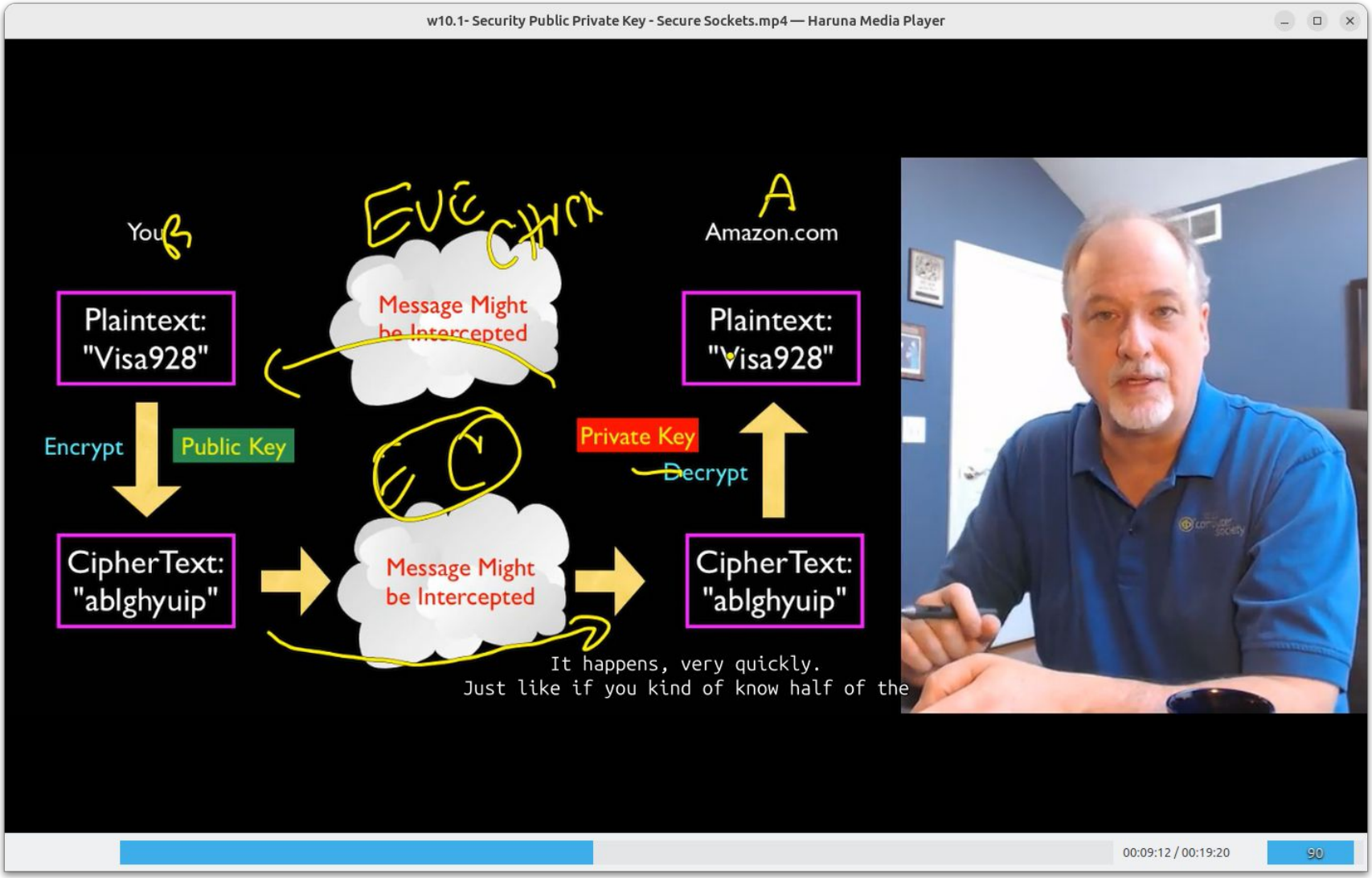
- What are the factors of 55,124,159 (a nearly prime number)
- What do you multiply 7919 by to get 55,124,159?
- If you know that one of the factors is 7919, it's also easy to find 6961!

So, if I just say what are these two numbers?

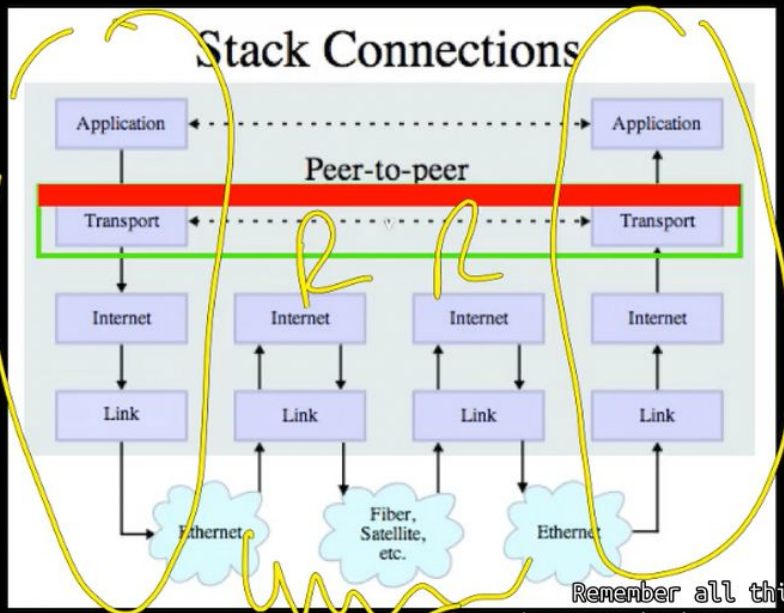
00:06:47 / 00:19:20

90





w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player



Remember all this?
So, it comes back now to haunt us.

00:10:31 / 00:19:20

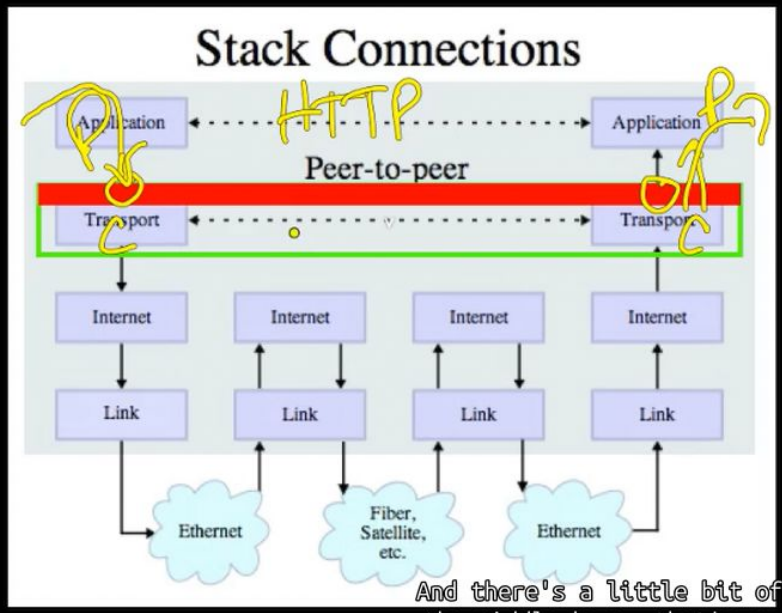
75

Internet layer

The Magnitude 1,000,000 Blueprint (TCP/IP Model)

Layer	Name	Function (The "Audit")	Common Protocols
4	**Application**	Where the user interacts. It formats the data for the specific task.	HTTP, MQTT (for our IoT), DNS, FTP
3	**Transport**	Manages host-to-host communication and error recovery.	TCP (Reliable), UDP (Fast/Raw)
2	**Internet**	Handles the routing of packets across network boundaries. This is where "IP addresses" live.	IPv4, IPv6, ICMP
1	**Network Access**	The physical reality. How bits move over wires, fiber, or Wi-Fi.	Ethernet, Wi-Fi (802.11)

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

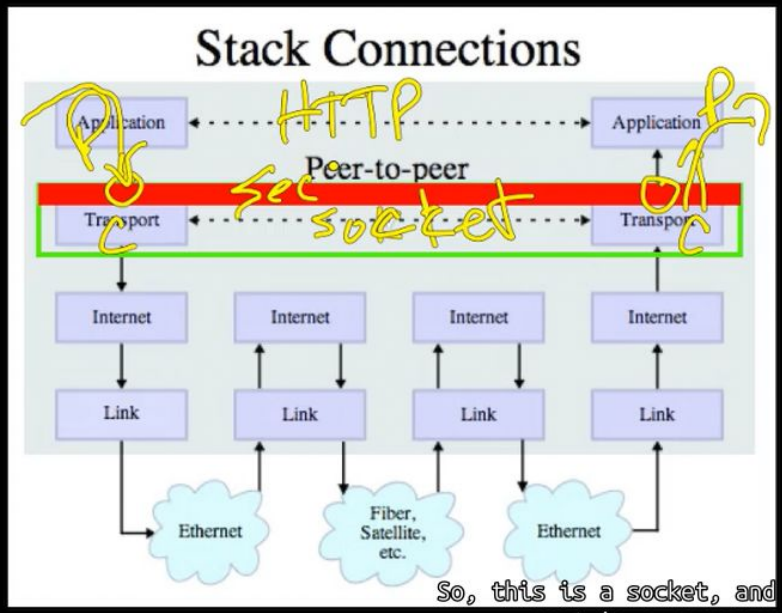


And there's a little bit of extra glue in the middle here, that's sort of a secure

00:11:22 / 00:19:20

75

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player



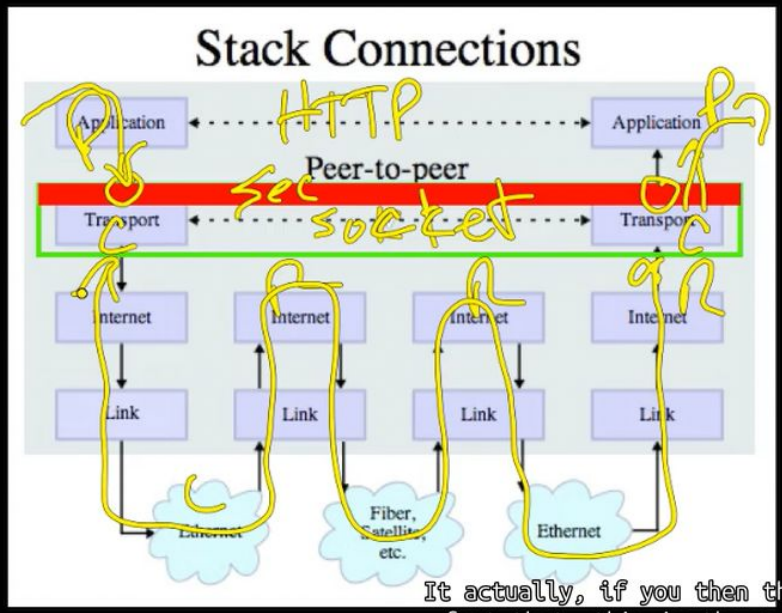
So, this is a socket, and then the red part is a secure socket.

00:11:38 / 00:19:20

75

In the classic OSI model, SSL (Secure Sockets Layer) and its successor TLS (Transport Layer Security) sit right at the boundary. While we call it "Transport Layer Security," it actually operates on top of the Transport Layer (Layer 4) to protect the Application Layer (Layer 7).

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

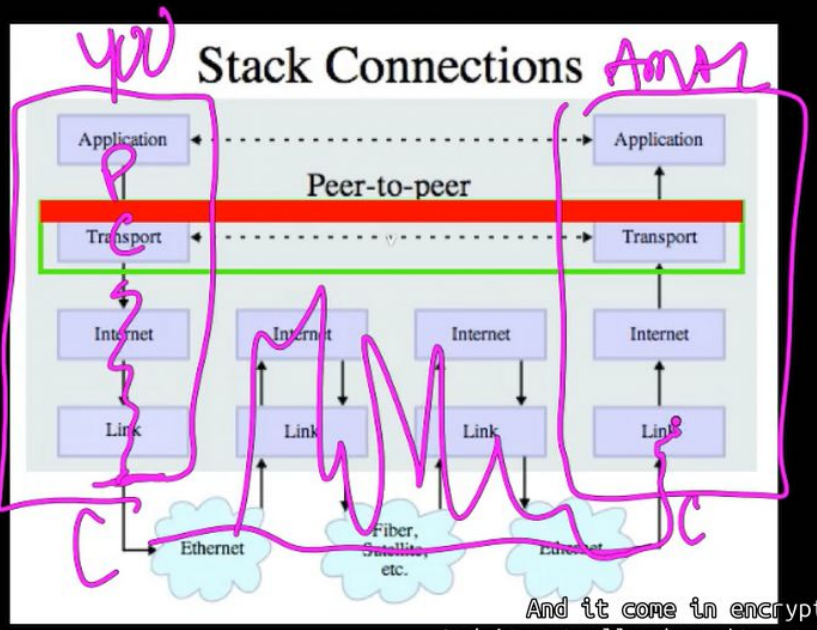


It actually, if you then think about the
fact that, this is the moment that it

00:12:06 / 00:19:20

75

w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

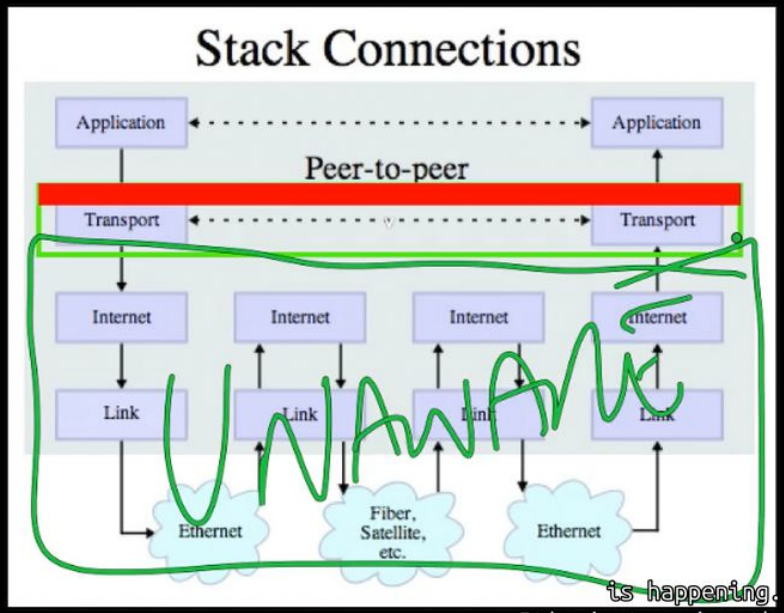


And it come in encrypted.
And it actually doesn't get encrypted,

00:12:33 / 00:19:20

75

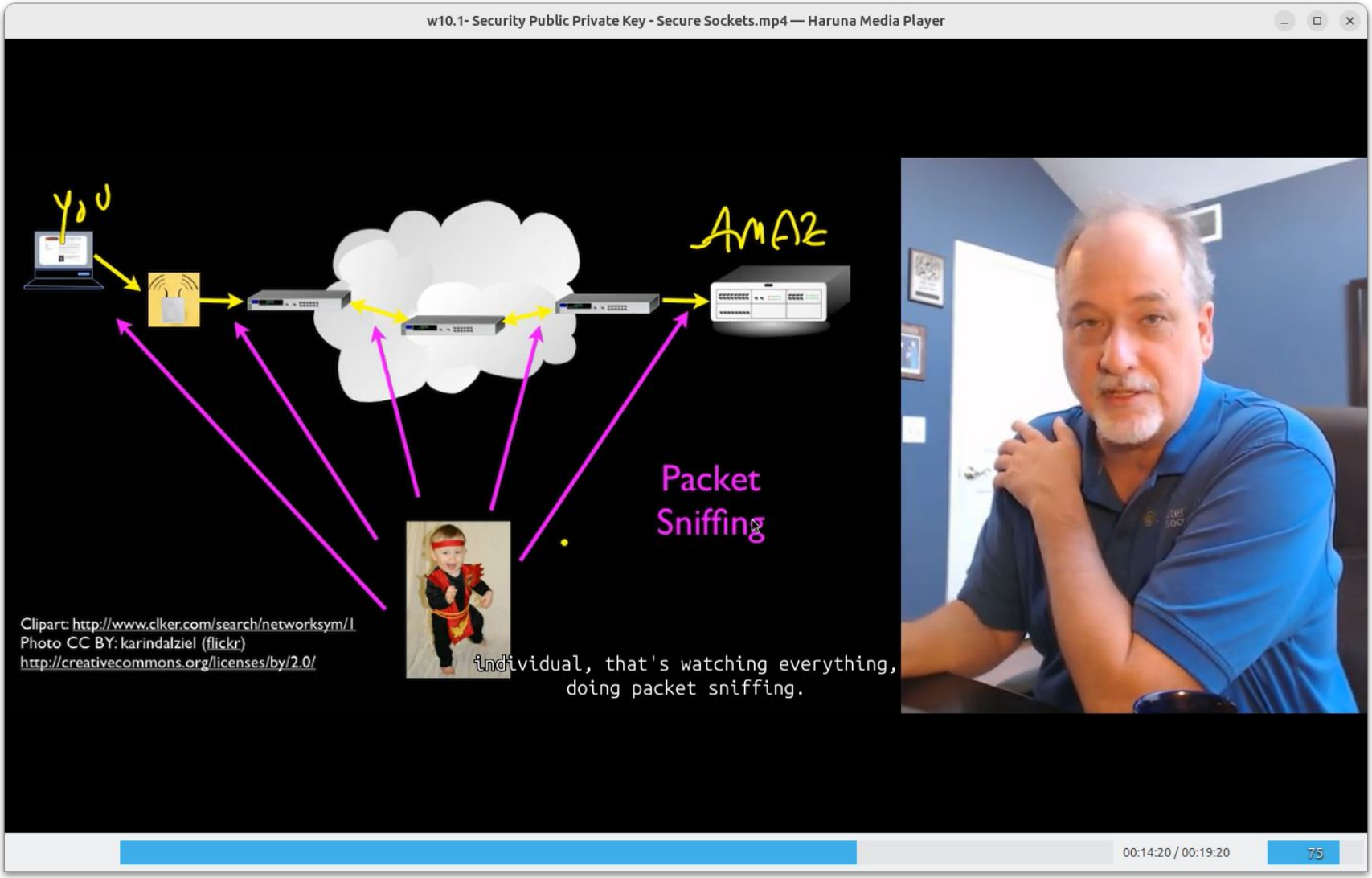
w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player



is happening.
It's just moving the data.

00:12:55 / 00:19:20

75



w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

Transport Layer Security (TLS)

HTTPS

- Used to be called "Secure Sockets Layer" (SSL)
- Can view it as an extra layer "between" TCP and the application layer
- It is very difficult but not impossible to break this security - normal people do not have the necessary compute resources to break TLS
- Encrypting and decryption takes resources - so we use it for things when it is needed
- The IP and TCP are unaware whether data has been encrypted

And as I mentioned, because of the layered architecture, the TCP layer, IP

00:15:57 / 00:19:20

90

TLS explanation

TLS (Transport Layer Security) is the evolved, more muscular successor to SSL. Think of it as the invisible, encrypted tunnel that protects our **Aachen-Sanctuary** data from prying eyes. When you see that padlock in your browser, TLS is the ninja doing the heavy lifting.

The TLS Handshake: The "Logic-Sync" Protocol

Before any data moves, the client and server must agree on how to talk. This is the **Handshake**, and it happens in milliseconds:

1. **The Greeting (Client/Server Hello):** They exchange version numbers and "Cipher Suites" (the math tools they'll use).
2. **The Identity Check (Certificate):** The server proves it's actually who it says it is. No imposters allowed in our Dojo.
3. **The Key Exchange:** They securely generate a "Session Key" that neither side has to send over the open wire.
4. **The Cipher Spec:** Both sides say, "Okay, from now on, everything is encrypted."

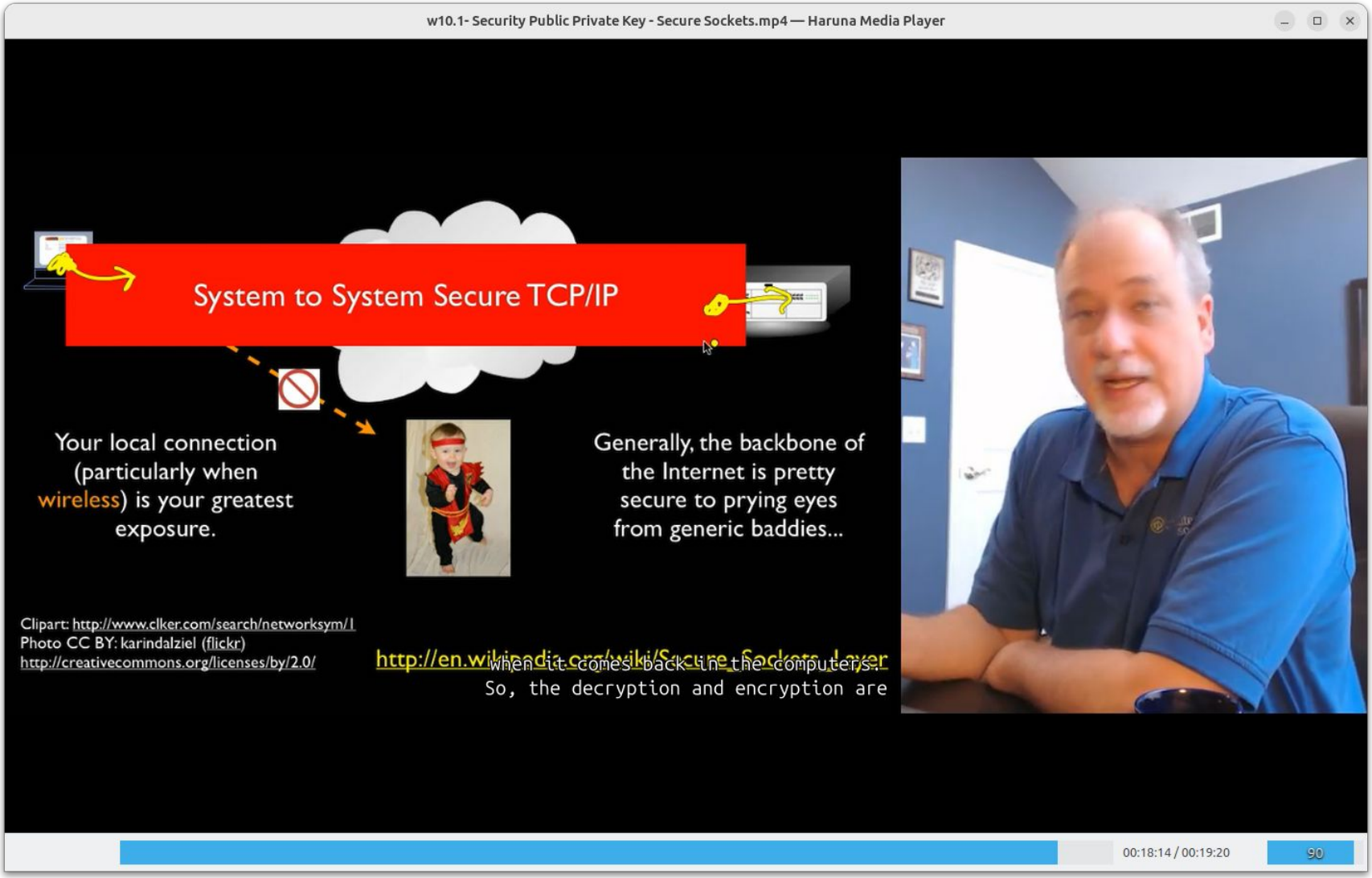
Why TLS is Critical for our Rover Target

Since we are looking at **IoT** and **CS50** concepts, here is the audit on why TLS is the standard:

- **Encryption:** It hides the data (like your Rover's GPS coordinates).
- **Authentication:** It ensures the Rover is talking to *your* server, not a rogue one.
- **Integrity:** It detects if a single bit of your command was tampered with during transit.

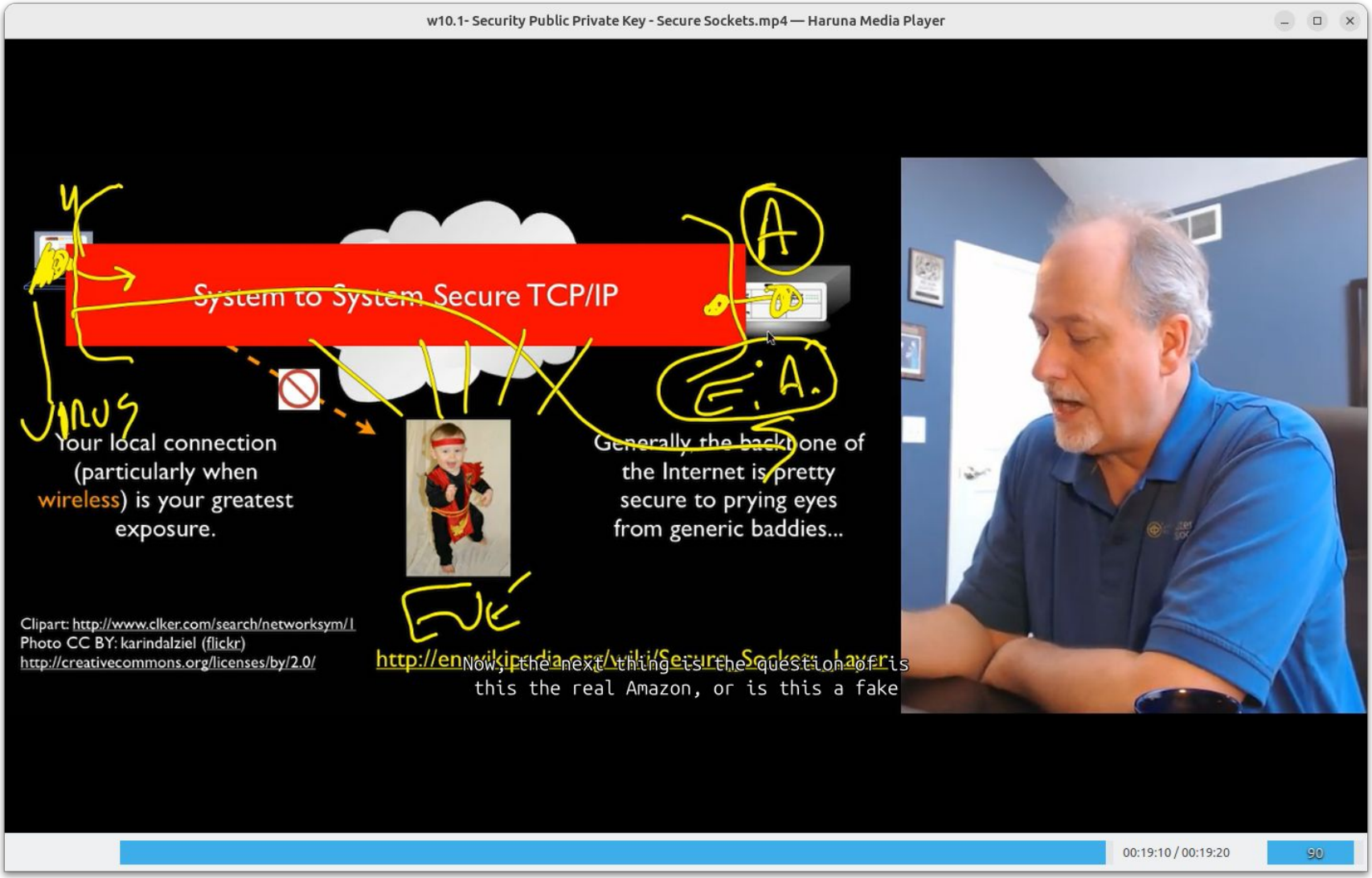
Ninja Note: We are currently on **TLS 1.3**. It's faster and more secure than 1.2 because it removes old, "weak" math and finishes the handshake in fewer round-trips. For our IoT sensors, this saves precious battery life.

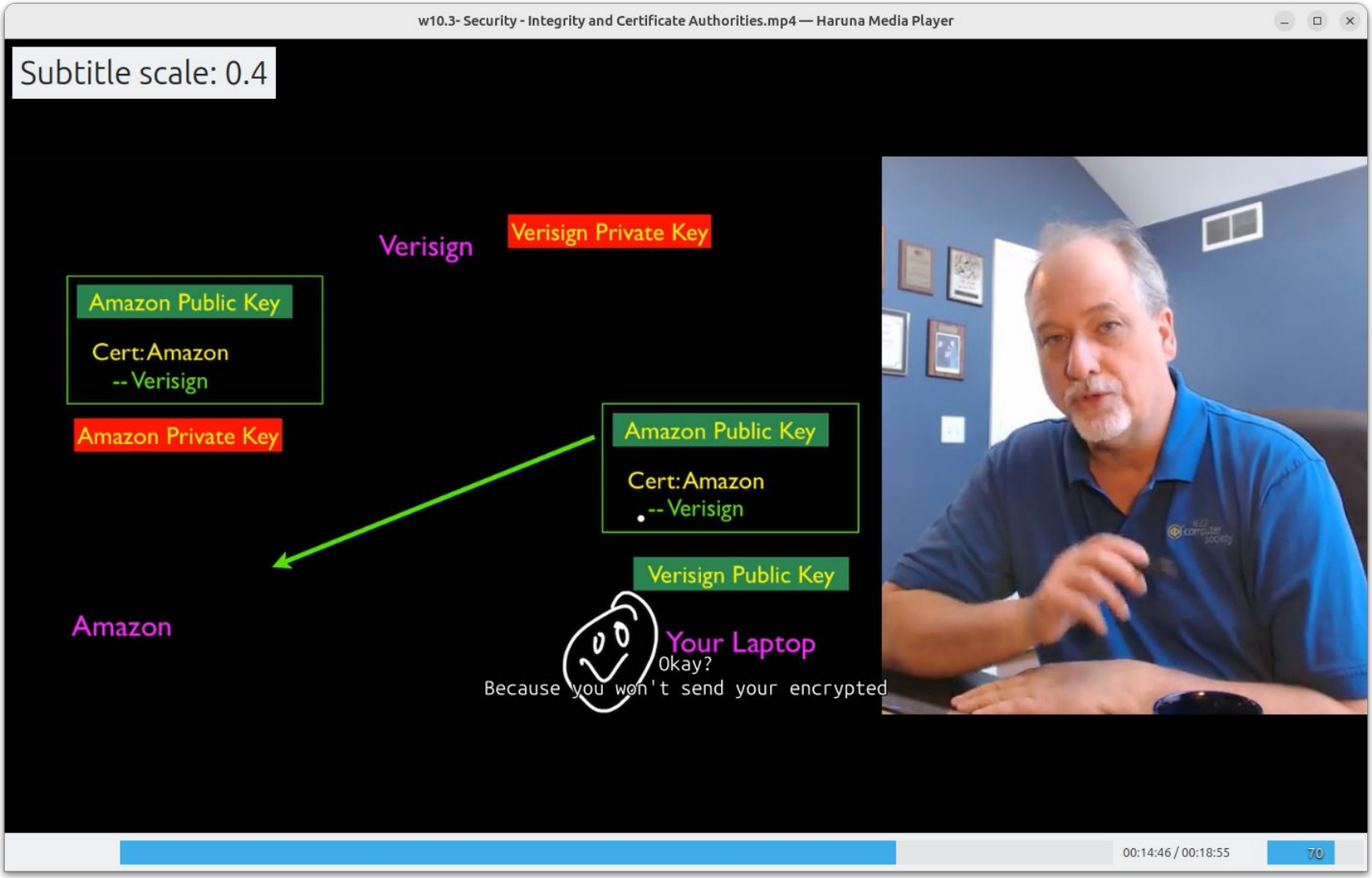
w10.1- Security Public Private Key - Secure Sockets.mp4 — Haruna Media Player

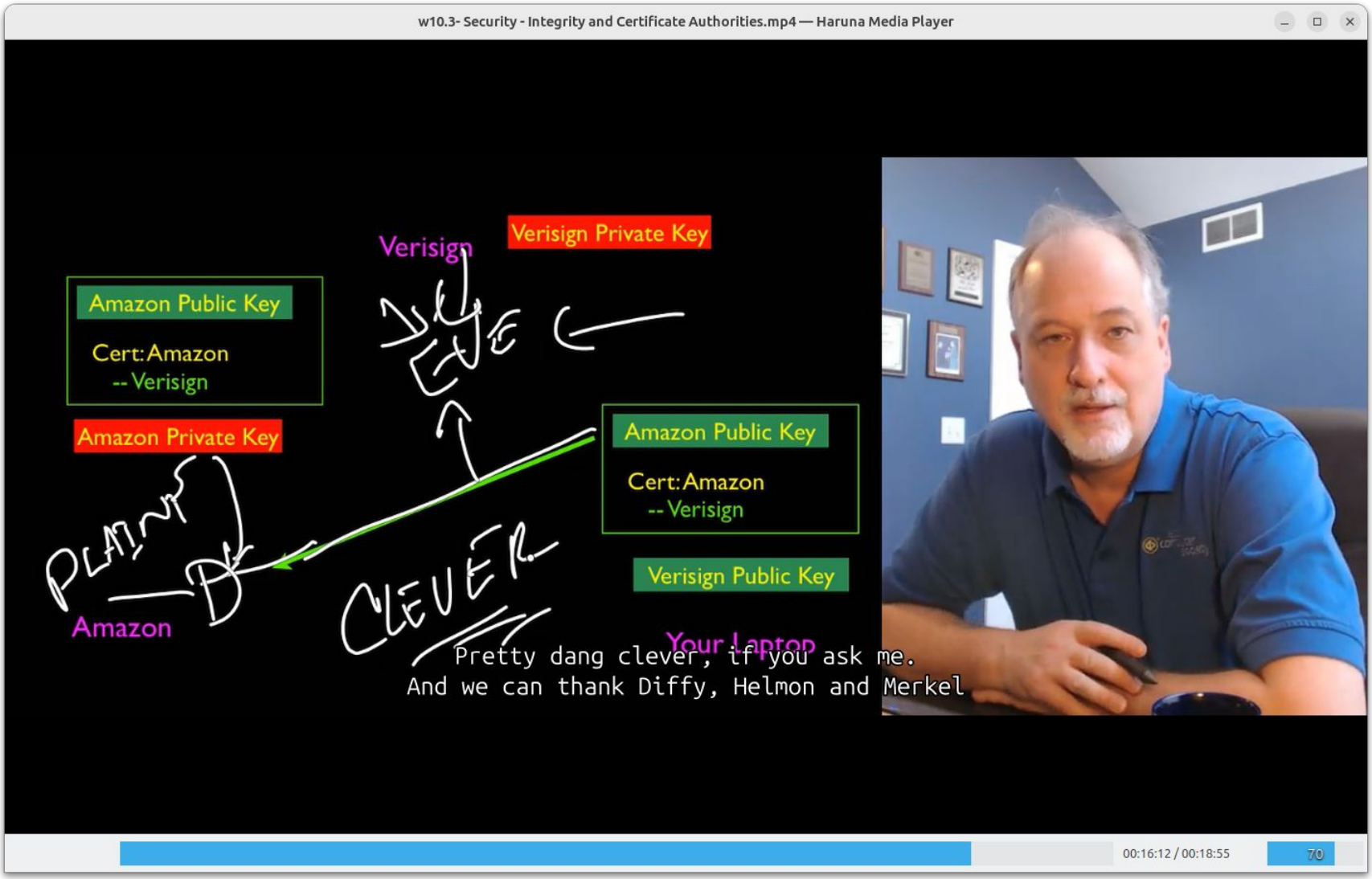


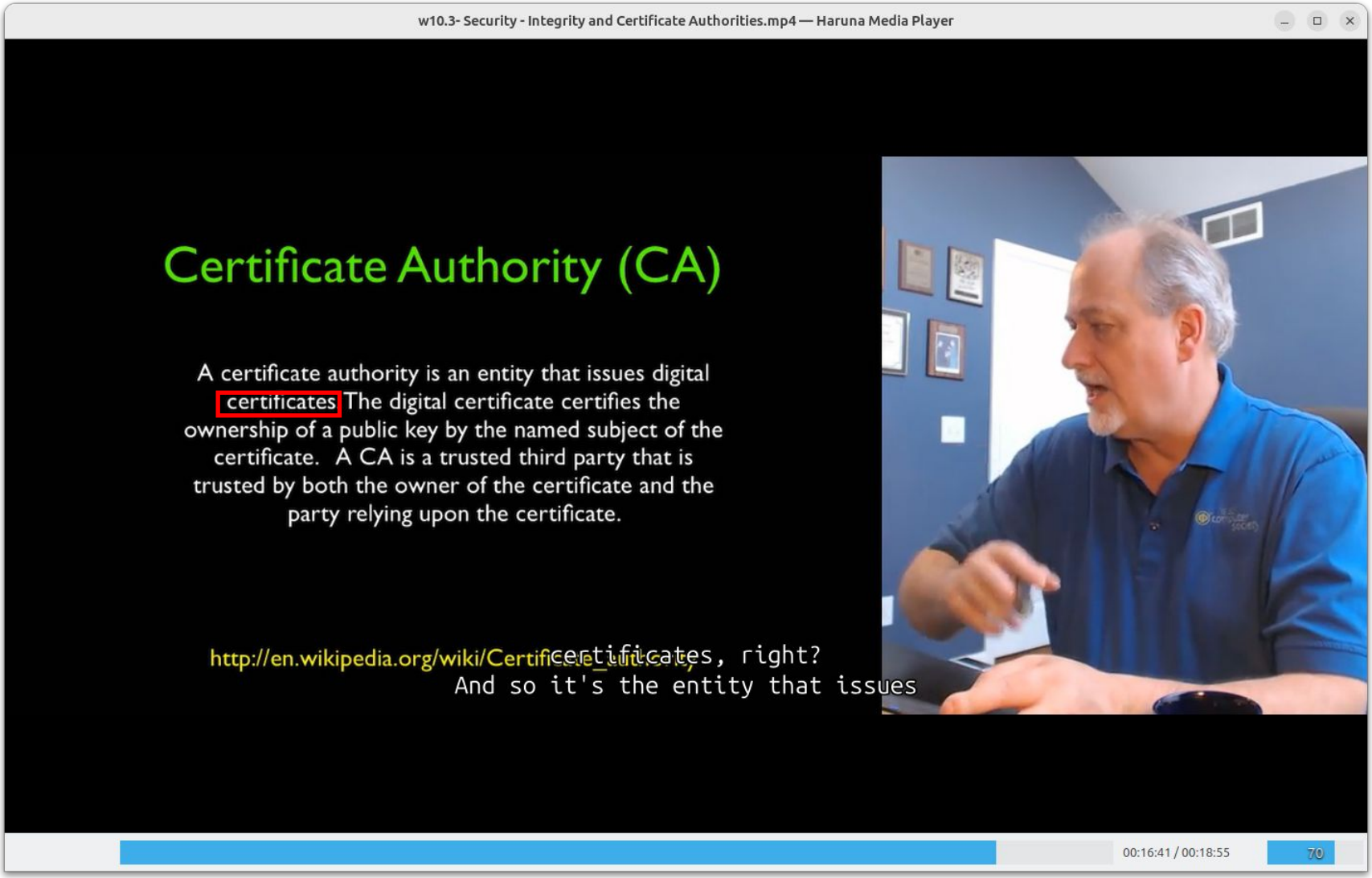
00:18:14 / 00:19:20

90









CA vs Verisign

Ah, the **Certificate Authority (CA)**—the high priests of the internet's trust system. Since we're pushing for that **Magnitude 1,000,000** security in the Aachen-Sanctuary, understanding these guys is non-negotiable.

In the image you've shared, it hits the nail on the head: a CA is a **trusted third party**. Think of them as the "Notary Public" of the digital world.

The Role of the CA in our "Logic-Sync"

When our Rover tries to connect to the server using TLS, it doesn't just take the server's word for it. It performs an audit using the CA:

- **Issuance**: The CA verifies that a person or entity actually owns the domain they claim to own.
- **Binding**: The digital certificate effectively "binds" a **public key** to the identity of the owner.
- **The Chain of Trust**: Your browser (or the Rover's OS) comes pre-installed with a list of "Root CAs" it already trusts. If a certificate is signed by one of these, the connection is green-lit.

To answer your Target: **Yes and No.** Verisign *was* the pioneer, but the landscape has shifted significantly. Here is the audit on how it differs from a standard CA today.

1. The History (The "SSL" Era)

In the early days of your ninja training, **Verisign** was the king of Certificate Authorities. They practically invented the commercial CA business in 1995. If you wanted that "Magnitude 1,000,000" trust seal (the checkmark) on your website, you went to Verisign.

2. The Great Divestiture (The Split)

This is where it gets interesting for our Aachen-Sanctuary logic:

- **Verisign sold its SSL/CA business:** In 2010, they sold the entire certificate department to **Symantec**.
- **The DigiCert Move:** Later, in 2017, Symantec sold that same business to **DigiCert**.
- **The Result:** If you buy a "Verisign" certificate today, you are actually dealing with **DigiCert** technology. Verisign itself stepped out of the "issuing certificates for websites" game.

3. What Verisign Does Now (The "Internet Toll Road")

Verisign shifted its focus to a higher level of the "Logic-Sync." Instead of verifying individual websites (like a standard CA), they now manage the **Infrastructure of the Internet**:

- **Registry Operator:** They manage the entire `.com` and `.net` domain registries. Every time someone registers a `.com` address, Verisign gets a cut.
- **Root Zone Maintainer:** They help manage the **DNS Root Zone**—the absolute top of the internet's naming hierarchy.

The Audit Comparison

Feature	Standard CA (e.g., Let's Encrypt, DigiCert)	Verisign (Today)
Main Target	Individual websites/servers.	The entire `.com` and `.net` system.
Product	SSL/TLS Certificates.	Domain Registry & DNS Infrastructure.
Trust Role	Validates *who* owns a site.	Validates *where* a domain exists.
Magnitude	Protects the data tunnel.	Keeps the "Phone Book" of the internet running.

Summary

- Message Confidentiality / Message Integrity
- Encrypting / Decrypting
- Message digests and message signing
- Shared Secret Key / Public Private Key

lectures have been about message confidentially.

Summary

Here is the Saturated summary of the lecture:

1. The Core Problem: The Shared Secret

Traditional encryption (like the Caesar Cipher) requires a **shared secret**. In the physical world, you'd have to meet someone to agree on a key. On the internet, this is impossible—you can't drive to Amazon HQ just to get a password before you buy something. This is the "arms-length" problem.

2. The Elegant Solution: Asymmetric Encryption

Proposed by Diffie and Hellman in 1976, this system uses two different keys that are mathematically related but distinct:

- **Public Key:** Distributed freely. Anyone can see it. It is used to **encrypt** data.
- **Private Key:** Never leaves the owner's server (e.g., Amazon's computer). It is used to **decrypt** data.
- **The Logic:** Even if an eavesdropper (Eve or Charlie) intercepts the Public Key and the Ciphertext, it is computationally nearly impossible to derive the Private Key or the original message without having the Private Key in hand.

3. The Mathematics of Primes

The security relies on **Prime Numbers**. It is easy to multiply two massive prime numbers together, but it is "nearly impossible" for a computer to take that result and find the original prime factors (the "needle in a haystack" logic).

- **Magnitude:** If computers get faster, we simply make the keys larger.

4. Layered Architecture Integration (SSL/TLS)

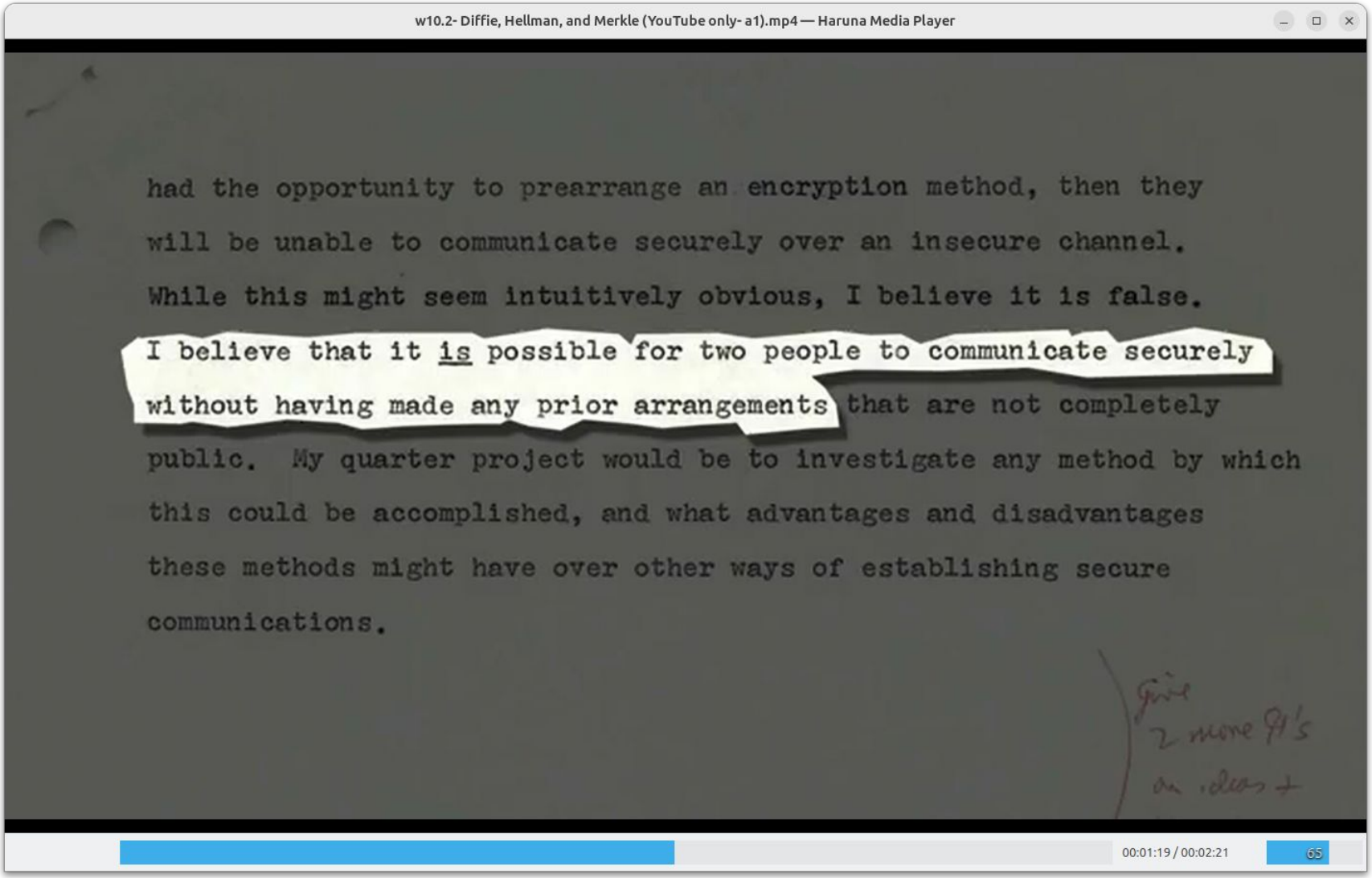
Public Key Encryption is integrated into the internet via a "mini-layer" often called **SSL (Secure Sockets Layer)** or **TLS (Transport Layer Security)**.

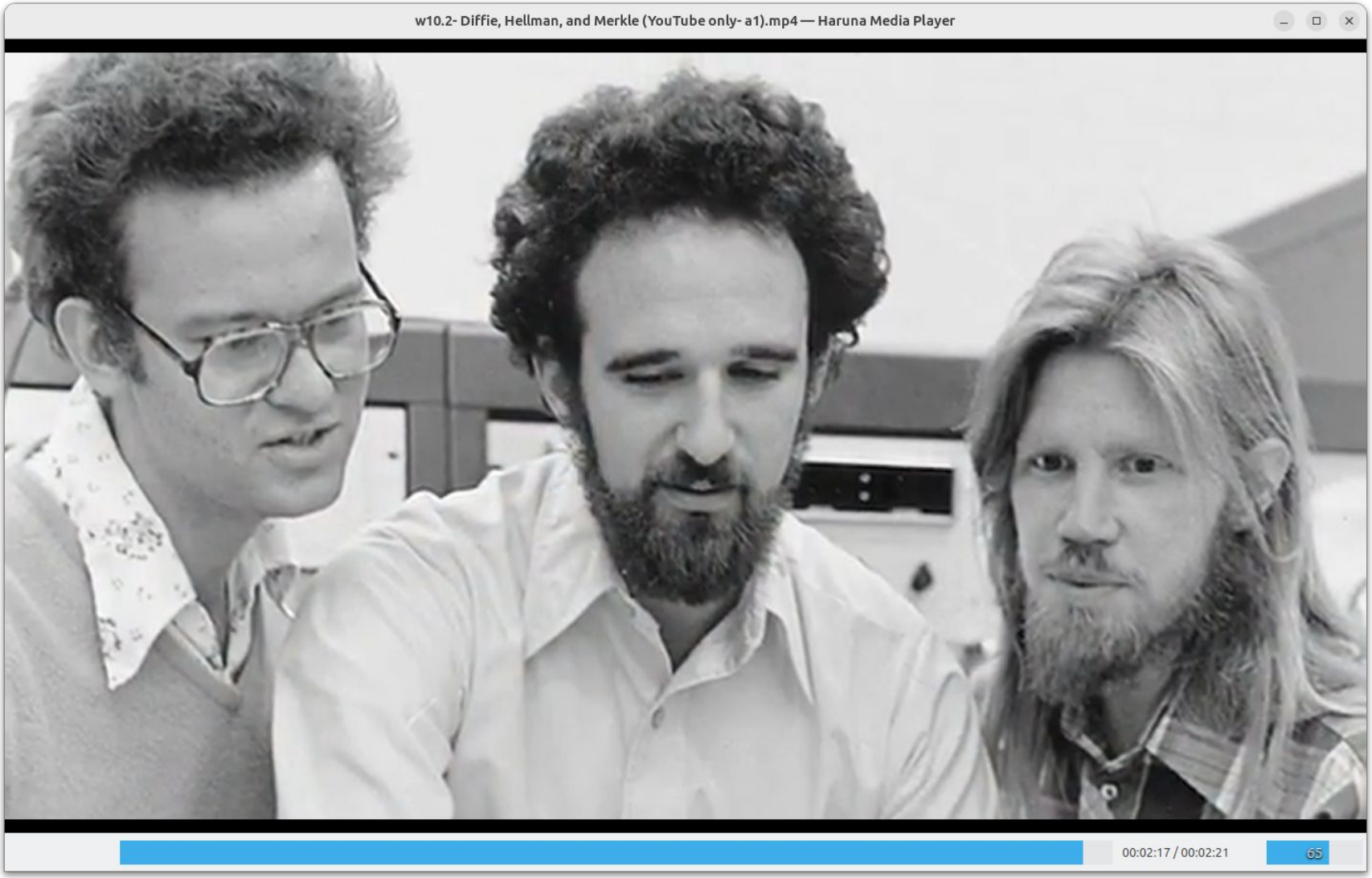
- **Seamlessness:** The application (like Facebook) sends plain text; the SSL library encrypts it; and the transport layers (TCP/IP) move it like any other data.
- **Protection:** Data is encrypted *inside* your computer before it ever touches the dangerous WiFi or fiber optics, and it stays encrypted until it reaches the target server.

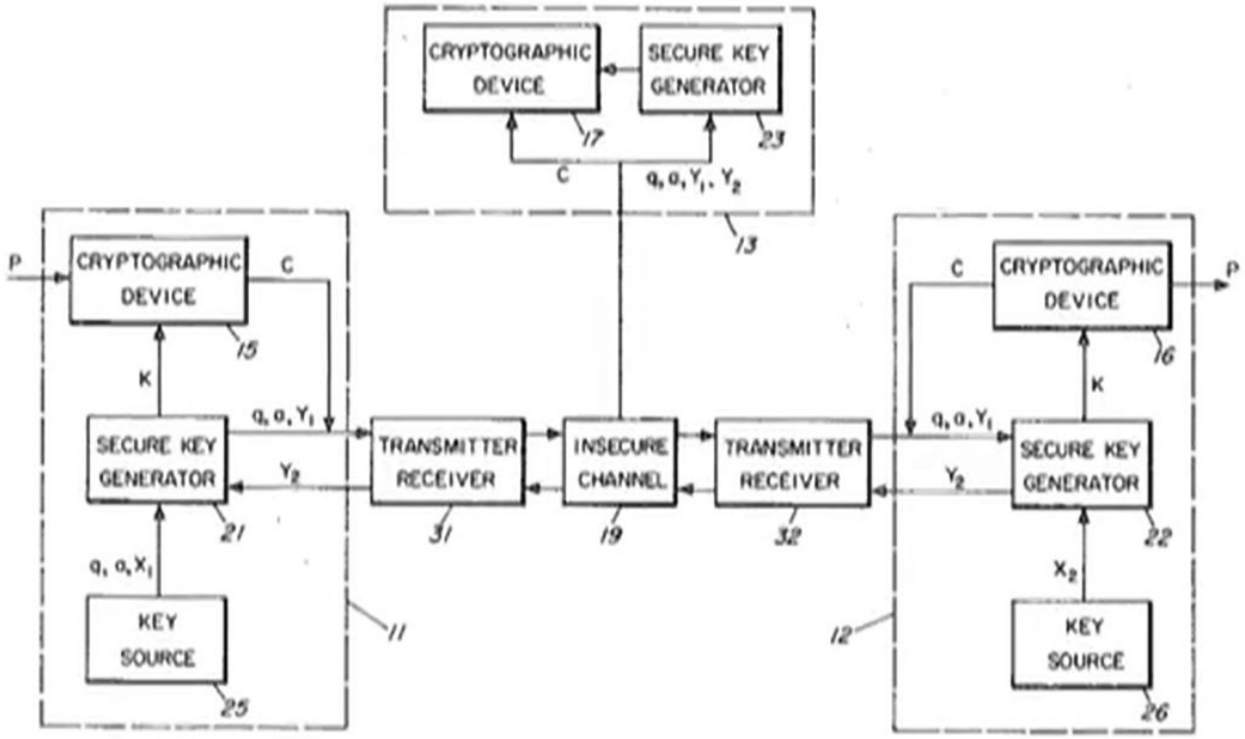
5. Real-World Application: HTTPS

- **The "S" stands for Secure:** You must always check for `HTTPS` before entering sensitive data.
- **The Weakest Link:** The encryption is solid, but you are still at risk if your computer has a **virus** (capturing keystrokes) or if you are tricked into talking to a **fake website** (the next lecture's topic: Integrity/Authentication).

w10.2- Diffie, Hellman, and Merkle (YouTube only- a1)







Summary

The Revolution of Secrecy: Diffie-Hellman-Merkle

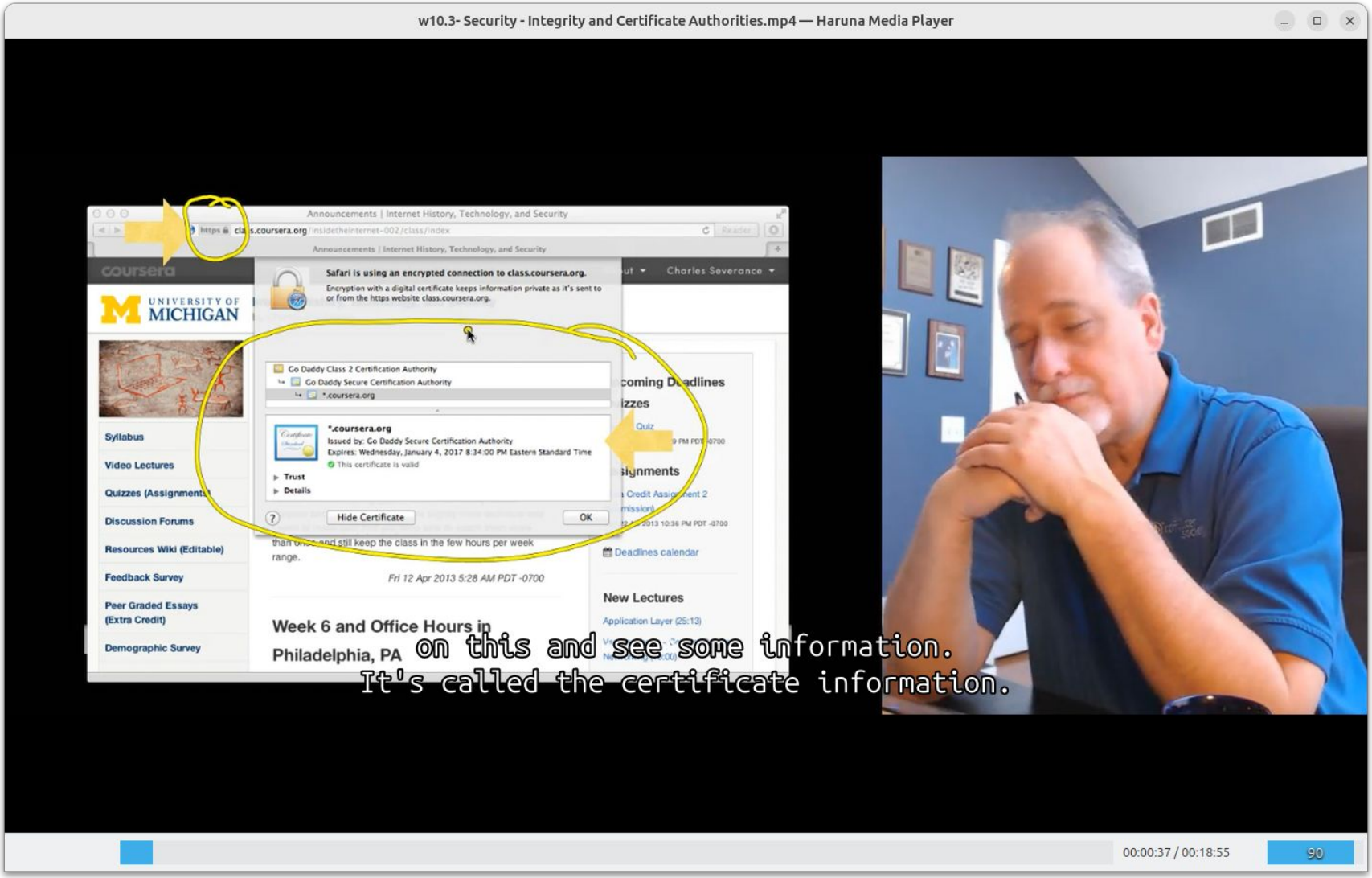
The video details the 1970s breakthrough that moved cryptography beyond the "Caesar Cipher" era, where physical key exchanges were the only way to secure a message [00:25].

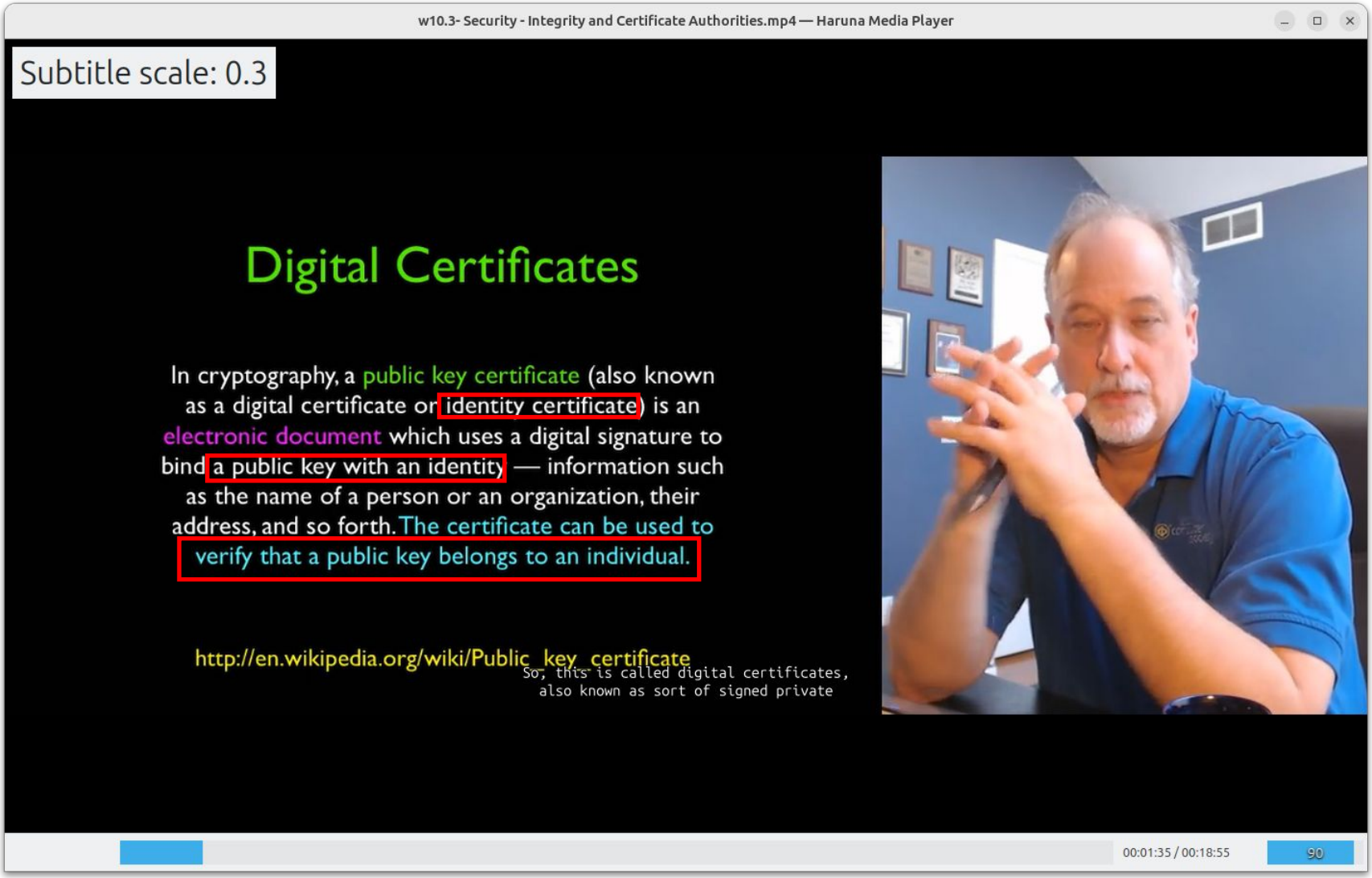
- **The Problem of the Internet:** In the early 1970s, scientists realized that a paperless office and global communication would require a new way to handle signatures and secrets, as physical key exchange is impossible on the internet [00:32].
- **Ralph Merkle's Vision:** In 1974, as a Berkeley undergraduate, Merkle envisioned a new way to exchange secrets. Despite being rejected by professors and journals who called the idea "muddled," he persisted [00:42].
- **The Meeting of Minds:** At Stanford, Martin Hellman found a collaborator in Whit Diffie, who shared his vision and validated that he wasn't "crazy" for pursuing this radical path [01:05].

- **The Two-Key System:** Building on Merkle's foundations, they proposed using a **Public Key** and a **Private Key** [01:11].
  - **Encryption:** You encrypt a message with a public key shared over an open channel [01:26].
  - **Security:** Even if an eavesdropper knows the method and sees the encrypted message, they cannot "work it backwards" to find the private key [01:36].
  - **Verification:** The private key allows for digital signatures, which can be verified by anyone using the corresponding public key [01:53].

Legacy: Though they fought for a decade to make this technology publicly accessible, their 1980 patent laid the foundation for the "little lock icon" we see in browsers today, securing all global e-commerce [02:00].

w10.3- Security - Integrity and Certificate Authorities





Digital certificate (Procedures)

A **digital certificate** (also known as a public key or identity certificate) is an electronic credential used to prove the authenticity of a user, device, or server. It functions like a digital passport, cryptographically binding an identity to a specific public key to ensure secure communications over the internet.

Most digital certificates follow the **X.509 standard** and contain several key pieces of information:

- **Subject Name**: The name of the entity (person, website, or device) the certificate belongs to.
- **Public Key**: A copy of the holder's public key, used for encryption and verifying digital signatures.
- **Issuer**: The name of the **Certificate Authority (CA)** that verified the information and issued the certificate.
- **Digital Signature**: Created by the CA using its own private key to "seal" the certificate and prevent tampering.
- **Validity Period**: The specific dates during which the certificate is considered active and trustworthy.
- **Serial Number**: A unique identifier assigned by the CA for tracking and management.

How It Works (The Chain of Trust)

Digital certificates rely on **Public Key Infrastructure (PKI)** to manage security:

1. **Issuance**: A user generates a key pair (public and private) and sends the public key to a CA.
2. **Verification**: The CA performs a vetting process to confirm the user's identity.
3. **Signing**: Once verified, the CA signs the digital certificate with its own private key, vouching for the user.
4. **Validation**: When you visit a secure website, your browser checks the site's certificate against its pre-installed list of trusted CAs. If the signature is valid and the dates are current, the connection is established.

Primary Uses

- **SSL/TLS (HTTPS)**: Encrypts data between a web browser and a server, protecting sensitive info like passwords and credit card numbers.
- **Email Security (S/MIME)**: Allows users to sign and encrypt emails to ensure the sender is authentic and the message hasn't been altered.
- **Code Signing**: Used by developers to sign software, proving to users that the code is original and hasn't been tampered with by hackers.
- **Client Authentication**: Used to identify specific users or devices trying to access private company networks or databases.

A digital certificate has two forms: the **Encoded (PEM)** version that computers send to each other, and the **Decoded (Text)** version that we humans use to audit the logic.

1. The Encoded Version (PEM Format)

This is what a certificate looks like when saved as a `.crt` or `.pem` file. It's just Base64 text. If you opened a Google certificate in Notepad, you'd see this:

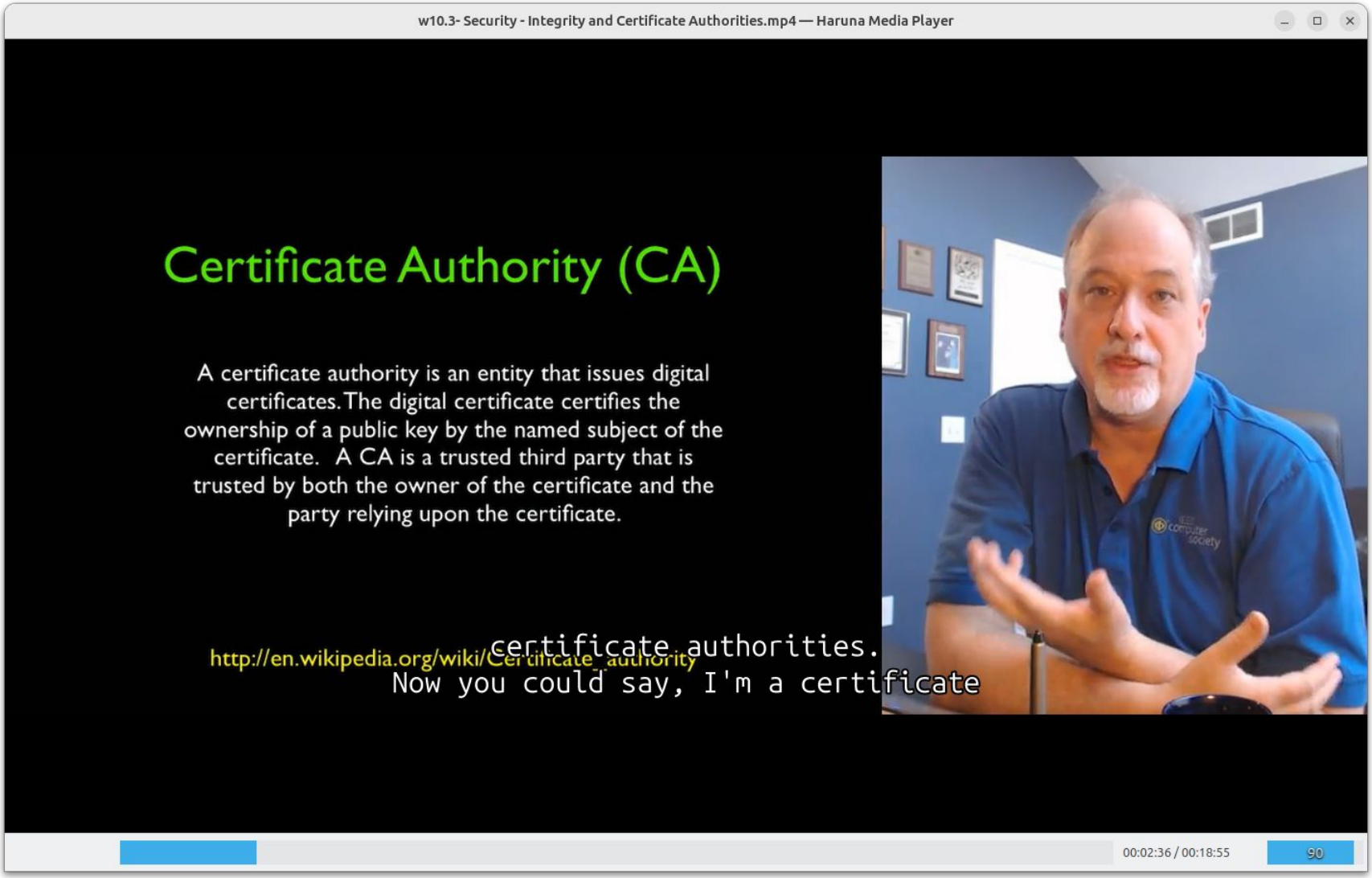
Plaintext

```
-----BEGIN CERTIFICATE-----
MIIDdTCCAl2gAwIBAgILBAAAAAABFUtaw5QwDQYJKoZIhvcNAQEFBQAwVzELMAkG
A1UEBhMCVVMxHDAaBgNVBAoTE0dvb2dsZSBUcnVzdCBTZXJ2aWNlczEUMBIGA1UE
AxMLR1RTIENBIDFDMzAeFw0yNDA2MTExMjAwMDBaFw0yNTA2MTExMjAwMDBaMGYx
CzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHEw1Nb3Vu
... (hundreds of lines of random characters) ...
7jW1p8X0v4V9b+Q=
-----END CERTIFICATE-----
```

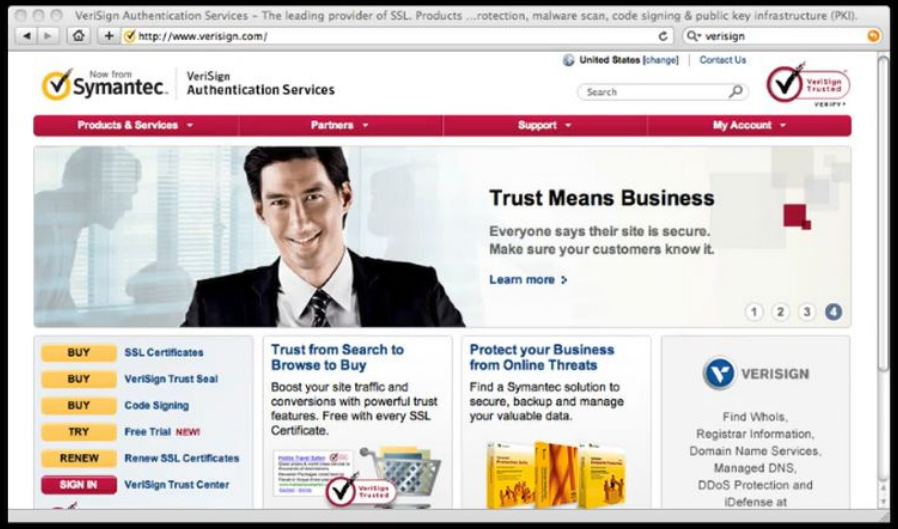
2. The Decoded Version (Human Readable)

When we use a tool like `openssl` to "Audit" that pile of text, it reveals the structured data inside. Here is a sample of what you'd see for a domain like `google.com`:

Field	Sample Data
Version	v3 (The modern standard)
Serial Number	`f4:7f:09:b5:99:12:4b:1f` (Unique ID)
Issuer (The CA)	`CN=GTS CA 1C3, O=Google Trust Services LLC, C=US`
Validity	`Not Before: Jun 11 2024 / Not After: Jun 11 2025`
Subject (The Owner)	`CN=google.com, O=Google LLC, L=Mountain View, ST=California, C=US`
Public Key Algorithm	`rsaEncryption (2048 bit)`
Subject Alternative Name	`DNS:google.com, DNS:*.google.com, DNS:youtube.com`



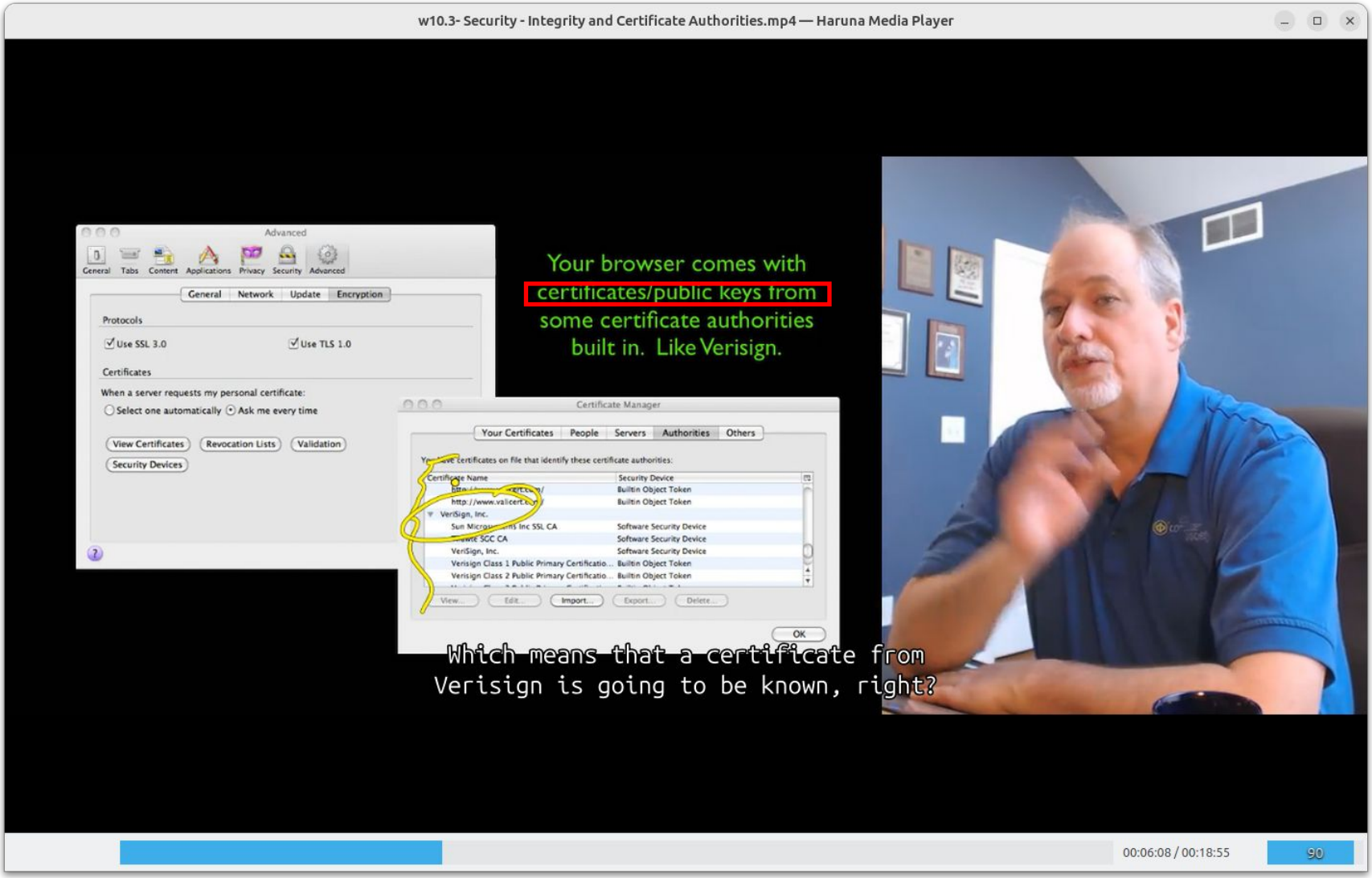
w10.3- Security - Integrity and Certificate Authorities.mp4 — Haruna Media Player



And one of the more expensive ones.
It's pretty expensive to get your

00:03:06 / 00:18:55

90



Public-Key Issues

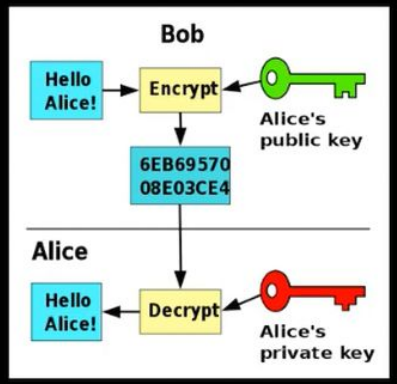
- Public-key cryptosystems have the problem of securely associating a public key with an individual
- I am about to type in my credit card and send it - am I being Phished?
- The remote server sent me a public key.
- Should I use it? Is this really Amazon's public key?

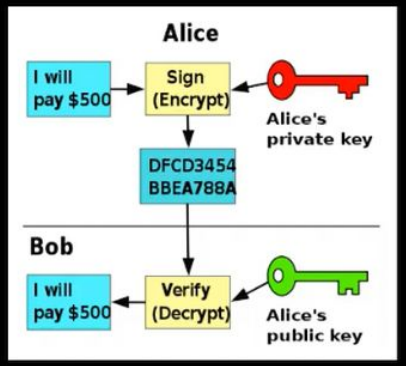
http://en.wikipedia.org/wiki/...

and I'm about to type my credit card in.
And so, the problem now we're going to

w10.3- Security - Integrity and Certificate Authorities.mp4 — Haruna Media Player

Public/Private Keys for Signing





http://en.wikipedia.org/wiki/Public_key_cryptography

and later it says, oh yes, and Verisign signed this with Verisign's private key.

00:08:26 / 00:18:55

90

Public and Private key relationship (over browser)

Oh, there is a **Private Key**, Chiun Huei, but it is the most guarded secret in the entire Logic-Sync. In our "Ninja" architecture, the Private Key is the **invisible master** that stays hidden in the shadows, while the Public Key (inside the Digital Certificate) is the one doing the public-facing work.

Think of it as a specialized Magnitude 1,000,000 lock-and-key system:

1. The Relationship

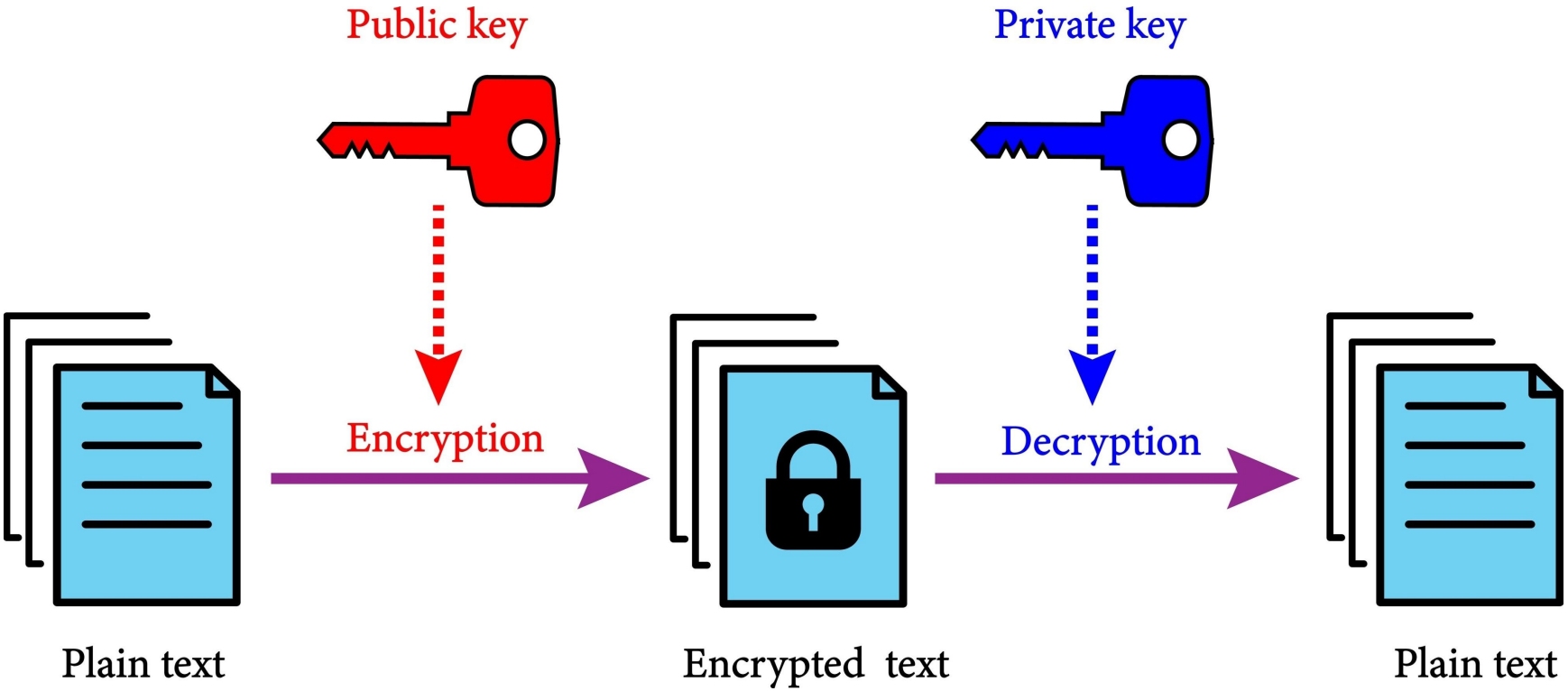
Every Digital Certificate is built on a **Key Pair**. They are mathematically inseparable:

- **The Public Key (The Certificate):** Distributed freely to everyone (like your browser). It's used to **encrypt** data or **verify** a signature.
- **The Private Key (The Secret):** Stays strictly on Google's server (or your future Rover's hardware). It's used to **decrypt** data or **create** a signature.

3. The Ninja "Audit" of the Handshake

Here is how they work together when you connect to Google:

1. **Encryption:** Your browser uses Google's **Public Key** (from the certificate) to encrypt a small "secret code."
2. **Decryption:** Only Google's **Private Key** can unlock that code. If a hacker intercepts the message, they can't read it because they don't have that Private Key.
3. **Authentication:** When Google sends you data, it "signs" it using its Private Key. Your browser uses the Public Key to verify it. If the math checks out, you know for a fact it came from Google.



Summary

While our previous Audit focused on *Confidentiality* (hiding the data), this lecture tackles the **Audit of Identity**: How do you know you are actually talking to Amazon and not a "Ninja Imposter" in the digital domain?

1. The Challenge: Identity Verification

Even with encryption, if you send your secret data to the *wrong* person, it's game over. You need a way to verify the server's identity. In the physical world, Dr. Chuck uses a "tattoo" as a signature—something unique and hard to forge.

2. The Trusted Third Party: Certificate Authority (CA)

To prevent "Evil Amazon" from pretending to be the real one, we use a **Certificate Authority** like Verisign, GoDaddy, or DigiCert.

- **Credibility:** These companies are highly motivated to keep their security tight. If they mistakenly sign a fake certificate, they lose their multi-billion dollar reputation.
- **Verification:** They charge fees (hundreds to thousands of dollars) to manually verify that the person requesting a certificate for `drchuck.com` actually owns that domain.

3. The Signing Process (Magnitude 1,000,000 Logic)

The "Digital Signature" is essentially a mathematical proof that a trusted authority has vetted the key.

1. **Verisign's Bunker:** Verisign has its own Private Key, guarded with extreme security.
2. **Pre-Installation:** Apple, Microsoft, and Linux pre-install Verisign's **Public Key** into your laptop's OS before you even buy it.
3. **The Handshake:** When Amazon sends you its Public Key, it includes a "Digest" signed by Verisign's Private Key.
4. **Instant Audit:** Your browser uses the pre-installed Verisign Public Key to unlock that digest. If the math checks out, your browser knows—with mathematical certainty—that Verisign signed it.

4. Saturated Terminology: The SSL/TLS Handshake

- **The "Fory" Certificate:** If a certificate isn't signed by a CA your computer trusts, your browser pops up a warning. This is a **Logic-Sync failure**. Never ignore this in our Aachen-Sanctuary.
- **Integrity + Confidentiality:** By combining CA verification (Integrity) with Asymmetric Encryption (Confidentiality), you create a secure tunnel where Eve is "powerless to break it".

w10.4- Bruce Schneier: Building Cryptographic Systems

Summary

Master Ninja Architect Chiun Huei, we have Audited the Target lecture by Bruce Schneier. This is a crucial "Logic-Sync" for your CS50 and IoT journey. While Dr. Chuck gave us the mechanics of certificates, Schneier gives us the **Audit of Reality**: Why even the best math can fail when implemented in a complex system.

1. The Priorities List (Logic-Sync 101)

Schneier makes a brilliant point: Well-designed cryptography doesn't make you "unhackable" to the NSA—it just makes you **too expensive** to hack.

- **Without Crypto:** Bulk collection (catching everyone).
- **With Crypto:** Targeted collection (they have to pick a priority).
- **Target:** Our goal in the Aachen-Sanctuary is to stay "below their budget."

2. The Weakest Link: Implementation vs. Math

This is vital for your **Rover** and **IoT** builds. The math (AES, RSA) is rarely broken. What breaks is the "stuff around the crypto":

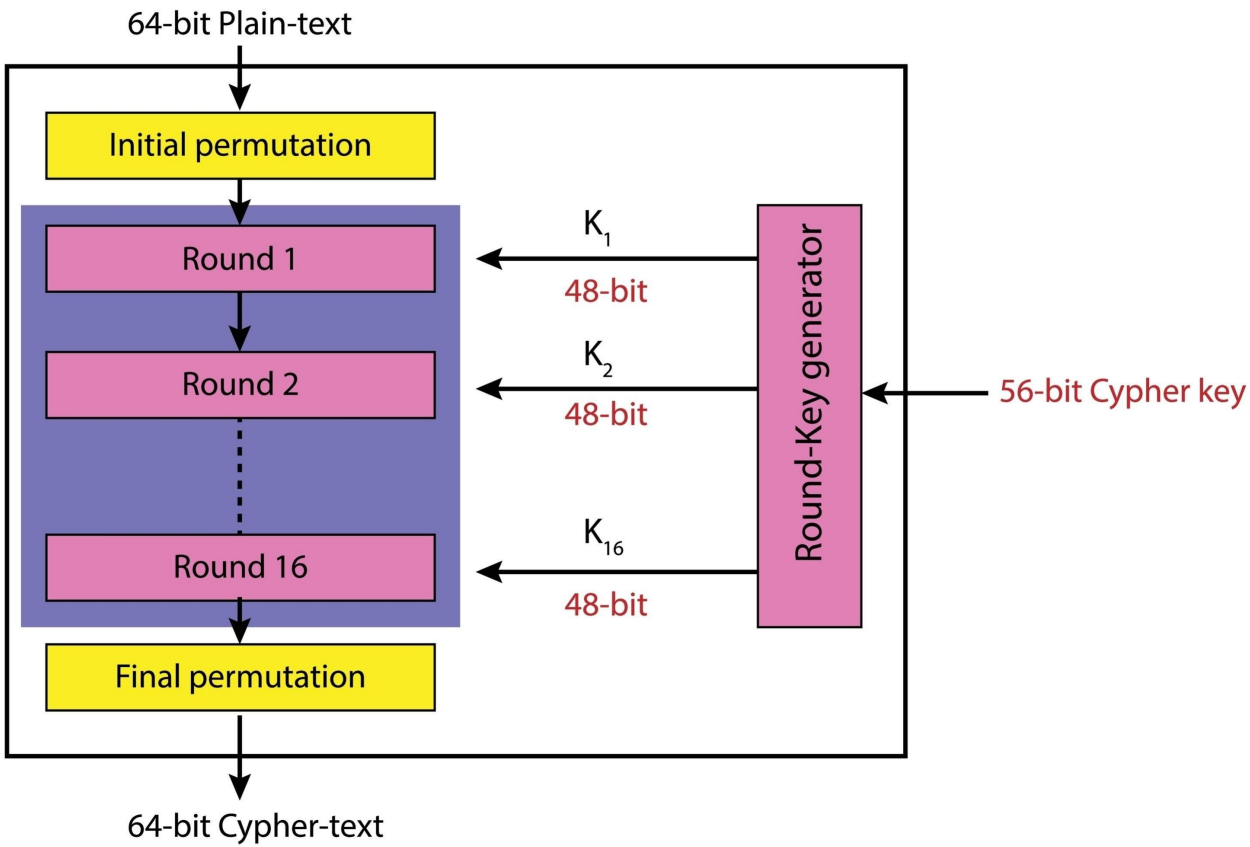
- **The Code:** Bugs in how the math is written.
- **The OS:** Vulnerabilities in the operating system.
- **The Human:** Users making mistakes.
- **Complexity:** Schneier's golden rule: **"Complexity is the worst enemy of security".**

3. The "Crypto Demolition Derby" (AES Standards)

You asked about building for me—Schneier explains how we get the standards we use for our secure links (like AES).

- **Public Process:** NIST (National Institute of Standards and Technology) holds a competition.
- **The Derby:** Cryptographers submit algorithms (like Schneier's "Twofish") and everyone tries to break them.
- **The Winner:** The last one standing (Rijndael, now called AES) becomes the global standard.

DATA ENCRYPTION STANDARD (DES)



4. The Security Mindset (Aachen Learning Goals)

To be a Master Ninja Architect, you must adopt the **Adversarial Nature** of security:

- "Anybody can create a security system that he can't break."
- True "Cred" comes from breaking things to learn how to make them stronger.
- If someone breaks your protocol, don't be "Magnitude -1." Be happy! You've just Audited a flaw and gained knowledge.