CSE350-
NETWORK SECURITY
ASSIGNMENT NO. 4

Digitally Signed Degree Certificates

By –Angadjeet (2022071)
Arav amawate(2022091)

Introduction

Goal: On-demand issuance of PDF Degree Certificates & Grade Reports

Web Application:

- Built a responsive MERN-stack site with:
- Signup & Login pages (multi-factor auth)
- Dashboard for downloading signed PDFs

Inputs: Graduate's roll number + verified personal data

Outputs:

- Degree Certificate PDF
- Grade Report PDF

Security: Each document digitally signed and timestamped

Accurate GMT Timestamping

- Source: Public NTP servers (pool.ntp.org, time.google.com, time.cloudflare.com)
- Process: Query sequentially (5 s timeout each), parse response into UTC
- Embedding: “YYYY-MM-DD HH:MM:SS GMT” on all PDFs
- Rationale: Uniform, traceable issuance time; upgradeable to NTS for encrypted sync

```python
import ntplib
import datetime

def get_gmt_datetime():
    """
    Attempts to fetch UTC from several NTP servers.
    Raises an exception if all attempts fail.
    """
    client = ntplib.NTPClient()
    servers = [
        "pool.ntp.org",
        "time.google.com",
        "time.cloudflare.com"
    ]
    last_exc = None

    for host in servers:
        try:
            # 5-second timeout per request
            response = client.request(host, version=3, timeout=5)
            return datetime.datetime.utcfromtimestamp(response.tx_time)
        except Exception as e:
            last_exc = e

    raise last_exc
```

Multi-Factor Access Control

- What you know: Roll number, Date of Birth, Pincode
- What you have: Password set at signup
- What you receive: One-Time Password (OTP) via Gmail
- Outcome: Only the legitimate graduate can authenticate and download documents

```python
SENDER_EMAIL    = "arav22091@iiitd.ac.in"
SENDER_PASSWORD = "rkpu gzzq nmig edke"

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT   = 587

def send_otp_email(recipient: str, otp: str):
    """
    Sends an OTP via SMTP using configured sender credentials.
    """
    subject = "Your One-Time Password"
    body    = f"Your OTP code is: {otp}\nPlease do not share it."
    msg     = MIMEText(body)
    msg["Subject"] = subject
    msg["From"]    = SENDER_EMAIL
    msg["To"]      = recipient

    with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
        server.starttls()
        server.login(SENDER_EMAIL, SENDER_PASSWORD)
        server.send_message(msg)
```

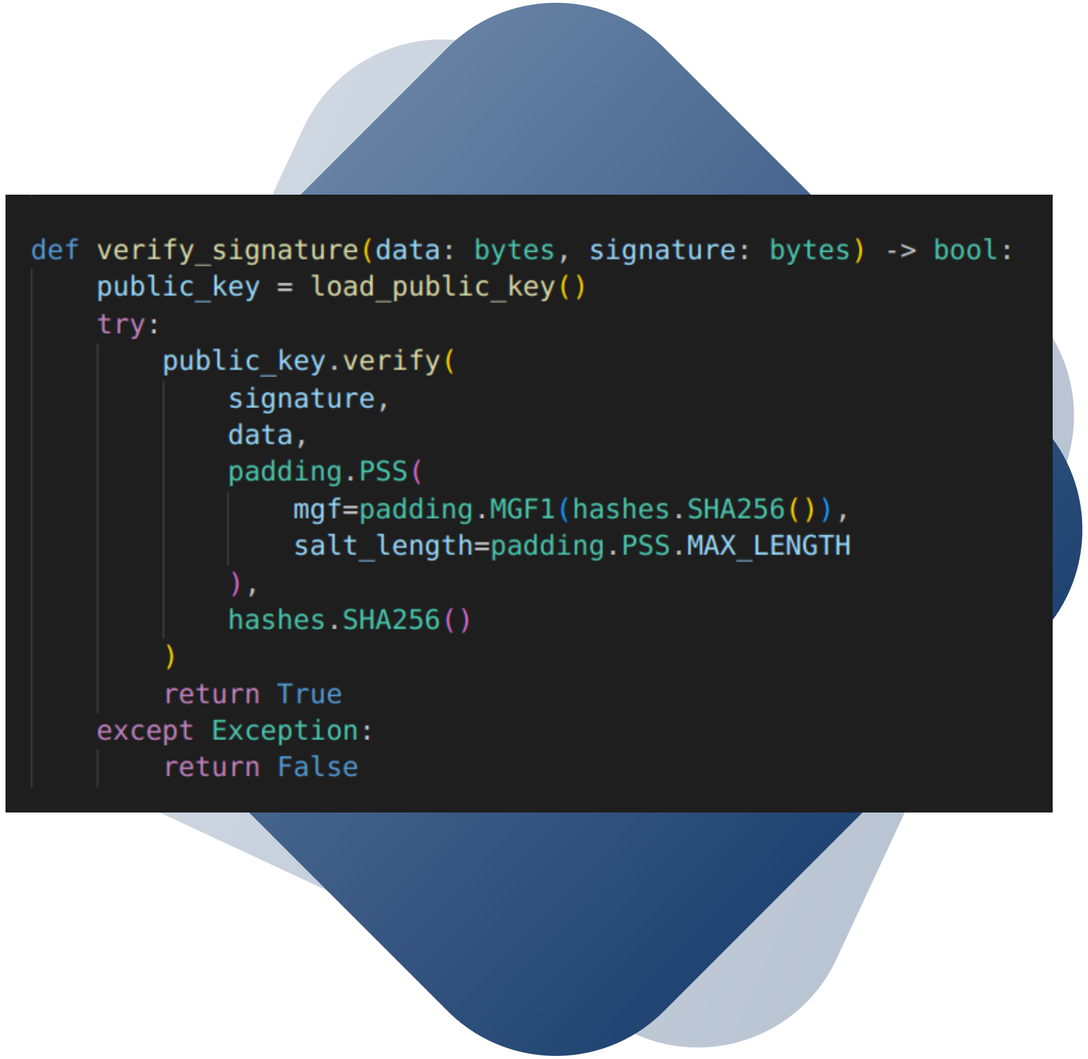
Traceability via Watermarks

- Watermark content: “Login Time: YYYY-MM-DD HH:MM:SS GMT”
- Placement: Semi-transparent, diagonal across each PDF page
- Benefit: Unique per-session fingerprint for forensic tracing of any circulated copy

```python
def generate_degree_certificate(user: dict, output_dir: str = "output") -> str:
    output_dir = os.path.join(BASE_DIR, "output")
    os.makedirs(output_dir, exist_ok=True)
    dt_issue = get_gmt_datetime().strftime("%Y-%m-%d %H:%M:%S GMT")
    watermark = f"Login Time: {user['login_time'].strftime('%Y-%m-%d %H:%M:%S GMT')}"

    buf_cert = io.BytesIO()
    c = canvas.Canvas(buf_cert, pagesize=LETTER)
    w, h = LETTER

    c.setFont("Helvetica-Bold", 26)
    c.drawCentredString(w / 2, h - 80, "UNIVERSITY OF INDIA")

    c.setFont("Helvetica-Bold", 20)
    c.drawCentredString(w / 2, h - 120, "Official Degree Certificate")

    c.setFont("Helvetica", 14)
    text = c.beginText(100, h - 180)
    text.setLeading(20)
    text.textLine(f"This is to certify that {user['name']},")
    text.textLine(f"Roll Number: {user['roll_number']}, has successfully fulfilled the")
    text.textLine(f"academic requirements set by the {user['college']}.")
    text.textLine(f"Year of Graduation: {user['grad_year']}")
    text.textLine(f"Date of Issue: {dt_issue}")
    c.drawText(text)

    # Smaller watermark
    c.saveState()
    c.setFont("Helvetica", 36)
    c.setFillGray(0.95)
    c.translate(w / 2, h / 2)
    c.rotate(45)
    c.drawCentredString(0, 0, watermark)
    c.restoreState()

    c.showPage()
    c.save()
    buf_cert.seek(0)

    cert_bytes = buf_cert.getvalue()
    generate_keys()
    signature = sign_data(cert_bytes)
    sig_b64 = b64encode(signature).decode()

    buf_sig = _create_signature_page(sig_b64)

    reader_cert = PyPDF2.PdfReader(buf_cert)
    reader_sig = PyPDF2.PdfReader(buf_sig)
    writer = PyPDF2.PdfWriter()
    for p in reader_cert.pages:
        writer.add_page(p)
    for p in reader_sig.pages:
        writer.add_page(p)

    path = os.path.join(output_dir, f"{user['roll_number']}_degree_certificate.pdf")
    with open(path, "wb") as f:
        writer.write(f)
    return path
```

Public-Key Infrastructure

- Key generation: RSA 2048 on system initialization
- Private key: Securely held by university authorities
- Public key: Published via the university’s HTTPS endpoint
- Verification: PSS-SHA256 covers the full PDF, enabling public integrity check



```python
def verify_signature(data: bytes, signature: bytes) -> bool:
    public_key = load_public_key()
    try:
        public_key.verify(
            signature,
            data,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        return True
    except Exception:
        return False
```

Security Properties and Future scope

- Authentication: Multi-factor (personal data + password + OTP)
- Integrity: Digital signature detects any tampering
- Non-Repudiation: Signature by a single private key prevents repudiation
- Confidentiality (future): Optionally encrypt PDFs for per-recipient privacy
-

Future Enhancements:

- PDF encryption for confidentiality
- Adoption of NTS (NTP over TLS)

THANKS!