Project 1: Digitally Signed Degree Certificates

Authors:Arav Amawate(2022091) and Angadjeet Singh(2022071)
Date: 30-03-2025

Overview

We built a system that, upon receiving a graduate's unique roll number (and associated personal data), produces two downloadable PDFs: a Degree Certificate and a Grade Report. Each document embeds the student's name, roll number, college, year of graduation, and issue date; carries a semi-transparent watermark showing the exact login timestamp; and concludes with a digital signature generated by the university's RSA private key.

---

1. Time Sourcing (GMT Date & Time)

To ensure absolute accuracy of the issuance timestamp, the system queries standard NTP servers (`pool.ntp.org`, `time.google.com`, `time.cloudflare.com`) over UDP, verifying each response and falling back only after all attempts time out. This leverages widely-deployed, mutually authenticated NTP infrastructure. While UDP-based NTP itself is not encrypted, these public strata-1 servers are trusted and operated by major providers; for higher security, NTP TLS (NTS) could be adopted in production. The retrieved timestamp is converted to UTC and formatted as "YYYY-MM-DD HH:MM:SS GMT" before embedding in the PDF.

```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
```

---

2. Graduate-Only Access Control

Access is gated by multi-factor verification:

- **Personal data match**: roll number, date of birth, and home pincode must all coincide with the record stored in a protected database.
- **Password check**: graduates set a password at signup.
- **One-Time Password (OTP)**: after correct credentials, an email OTP is sent to the graduate's Gmail and must be entered within five minutes.

This layered approach ensures that knowledge of the roll number alone is insufficient—only the genuine graduate, possessing both personal details and email access, can retrieve the signed 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)
```

---

3. Traceability via Watermarks

Each PDF page carries a large, rotated watermark reading “Login Time: YYYY-MM-DD HH:MM:SS GMT.” Because this timestamp is unique per session, any shared copy can be traced back to the exact issuance moment (and thus the account session) under which it was downloaded. Should a document appear in unauthorized circulation, the embedded timestamp acts as a forensic fingerprint tying it to the specific download event.

---

4. Public-Key Distribution

Documents are signed with the university’s RSA private key (`private_key.pem`), and signature verification requires the accompanying public key (`public_key.pem`). That public key can be published on the university’s official website (via HTTPS) or embedded in client-side apps/distribution packages. Users or external verifiers simply load the public key and run a PSS-SHA256 verification over the PDF’s raw bytes, guaranteeing the signature’s provenance without ever exposing the private key.

```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
```

---

5. Relevant Security Properties

- **Authentication**: Ensured by passwords and email OTPs.
- **Integrity**: The RSA signature covers every byte of the PDF, so any post-signature alteration is immediately detectable.
- **Non-Repudiation**: Because the university's private key is exclusively held by authorized authorities, signed documents serve as irrefutable proof of issuance.
- **Confidentiality**: Not directly addressed by this system—PDFs are publicly downloadable once authenticated. If needed, one could encrypt the PDFs per-recipient (e.g. using public-key encryption) to prevent unauthorized reading.

Conclusion

This solution delivers tamper-evident, traceable academic credentials with strong assurances of authenticity and origin, while relying on standard, auditable cryptographic and time-synchronization practices. No graduate can forge or dispute a signature, and every download is uniquely watermarked for full accountability.

A