**Lecturer:** Nadim Kobeissi **Website:** https://appliedcryptography.page

## Final Exam: Real-World Cryptography

**Instructions**

- Closed book, closed notes.
- Each question part (e.g. Question 2, part (a)) can be answered with at most two paragraphs.
- Please only use the paper provided for you in the exam booklet.
- Please write legibly. Illegible answers will not be graded.
- No calculators, electronic devices, or internet access permitted.
- Phones, smartwatches, backpacks and all pocket items must be left in backpack at entrance.
- All electronic devices must be turned off.
- Removing staples from the exam booklet is not allowed.
- Do not write your name on the exam booklet.
- Leaving the final exam is not allowed until all students have completed the final exam.

## Topic 2.1: Transport Layer Security

1. (8 points) **TLS Handshake and Trust Model**

   A startup proposes simplifying TLS by having each website generate its own certificate and sign it with its own private key (self-signed certificates for everyone), eliminating certificate authorities entirely. They argue: "The certificate still binds the public key to the domain name, and the signature proves the server possesses the private key. This is simpler and removes the CA as a single point of failure."

   Identify the critical security flaw in this proposal. Describe a concrete man-in-the-middle attack that becomes possible, explaining step-by-step how an attacker on the network path could exploit this design. Then explain what specific property of the current CA-based system prevents this attack.

2. (10 points) **Historical TLS Attacks and Mitigations**

   Consider the following flawed TLS-like protocol fragment for CBC-mode decryption:

   1. Server receives encrypted record $C = \text{IV} \| C_1 \| C_2 \| \ldots \| C_n$
   2. Server decrypts to obtain $P_1 \| P_2 \| \ldots \| P_n$
   3. Server checks PKCS#7 padding in $P_n$: if the last byte is $k$, verify the last $k$ bytes all equal $k$
   4. If padding is invalid, server sends `PADDING_ERROR`
   5. If padding is valid, server checks MAC; if invalid, sends `MAC_ERROR`
   6. If both valid, server sends `OK`

   (a) Explain why distinguishing between `PADDING_ERROR` and `MAC_ERROR` creates a vulnerability. Describe how an attacker could iteratively modify ciphertext bytes and use the server's responses to decrypt a target ciphertext block without knowing the key.

   (b) In 2015, the FREAK attack exploited a protocol state machine flaw where an attacker could force a downgrade to "export-grade" 512-bit RSA. Suppose a client sends a `ClientHello` offering only strong cipher suites (e.g., `TLS_RSA_WITH_AES_128_CBC_SHA`). Describe step-by-step how a man-in-the-middle attacker could modify messages to cause the server to use export-grade RSA, even though the client never offered it. Your answer must explain: (1) what specific message the attacker modifies and how, (2) why the server accepts this modification, (3) why the client accepts the server's response despite not offering export ciphers, and (4) what fundamental verification was missing in vulnerable implementations that allowed this attack to succeed.

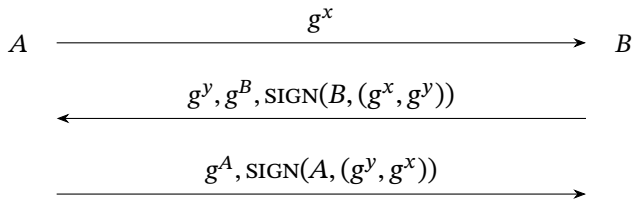
## Answer Pages — Topic 2.1

## Answer Pages — Topic 2.1

## Topic 2.3: Secure Messaging

1. (8 points) **Authenticated Key Exchange and SIGMA**

   A developer proposes the following "simplified SIGMA" protocol, removing the MAC:




   The developer argues: "The signature already covers both ephemeral keys, binding the identity to this specific session. The MAC is redundant."

   Construct a concrete identity misbinding attack against this simplified protocol. Your attack should show how a malicious party $M$ (who has their own valid identity key $g^M$) can cause $A$ to believe they completed a key exchange with $B$, while $B$ believes they completed it with $M$. Show each message $M$ sends or modifies and explain why the signatures still verify.

2. (8 points) **Signal Protocol Architecture**

   Consider a modified X3DH where signed prekeys (spk) are replaced with unsigned prekeys, making all prekeys equivalent:

   - Bob publishes: identity key $IK_B$, and a set of unsigned prekeys $\{PK_1, PK_2, ...\}$
   - Alice initiates by fetching $IK_B$ and one prekey $PK_i$ from the server

   Describe a concrete attack a malicious server (or attacker who compromises the server) can mount against this modified protocol that is *not* possible in standard X3DH. Your answer should specify: what the attacker substitutes, what key Alice derives, who can read Alice's first message, and why signed prekeys prevent this attack. Also explain why one-time prekeys do not need signatures despite this threat model.

## Answer pages — Topic 2.3

## Answer pages — Topic 2.3

## Topic 2.5: High-Assurance Cryptography

1. (16 points) **Implementation-Level Security: Side Channels and Constant-Time**

   (a) Consider this password verification function deployed on a server:

   ```
   bool verify_password(char *input, char *stored, int len) {
       for (int i = 0; i < len; i++) {
           if (input[i] != stored[i]) {
               return false;
           }
       }
       return true;
   }
   ```

   An attacker can submit login attempts and measure response times with microsecond precision. Describe a step-by-step attack to recover an 8-character password. Your answer must include: (1) the specific observation the attacker makes, (2) how many attempts are needed in the worst case to recover the first character (assume 62 possible characters: a-z, A-Z, 0-9), (3) the total worst-case attempts to recover all 8 characters, and (4) why this is dramatically fewer than brute force.

   (b) A developer "fixes" the above code as follows:

   ```
   bool verify_password_v2(char *input, char *stored, int len) {
       int result = 0;
       for (int i = 0; i < len; i++) {
           if (input[i] != stored[i]) {
               result = 1;
           }
       }
       return result == 0;
   }
   ```

   Explain why this fix is still insufficient on modern CPUs. Identify the remaining timing leak, and show a correct constant-time implementation using only bitwise operations and no conditional branches.

## Answer Pages — Topic 2.5

## Answer Pages — Topic 2.5

## Topic 2.6: Post-Quantum Cryptography

1. (8 points) **The Quantum Threat**

   Explain why the quantum threat timeline is genuinely less urgent for *authentication* and *signatures* compared to *key exchange* and *encryption*. What property distinguishes these use cases? However, describe one scenario involving signatures where retroactive quantum attacks still pose a serious threat, and explain what property of the signed data creates this vulnerability.

2. (8 points) **Post-Quantum Protocol Design and Deployment**

   A protocol designer proposes using *only* ML-KEM-768 (no hybrid with classical cryptography) for a new secure messaging application, arguing: "ML-KEM is now standardized by NIST. Using hybrid adds complexity, bandwidth overhead, and the classical component (X25519) will be broken by quantum computers anyway. Pure post-quantum is cleaner."

   Present two distinct arguments against this position, one commenting on cryptanalytic uncertainty, and one taking into account implementation maturity considerations. Finally, state the precise security guarantee of a hybrid like X-Wing in one sentence using "AND" or "OR" logic.

## Answer Pages — Topic 2.6

## Answer Pages — Topic 2.6

## Topic 2.8: Zero-Knowledge Proofs

1. (8 points) **Schnorr Protocol and Soundness**

   In the Schnorr protocol, the verifier sends a random challenge $c$ *after* receiving the prover's commitment $R$. Consider a modified version where the challenge is sent *before* the commitment (i.e., verifier sends $c$ first, then prover sends $R$ and $s$).

   Show a concrete attack where a prover who does not know the secret $x$ can produce a valid proof that passes verification. Your answer must: (1) specify exactly how the malicious prover computes $R$ and $s$ without knowing $x$, (2) demonstrate that these values satisfy the verification equation $g^s = R \cdot Y^c$, and (3) identify which security property of zero-knowledge proofs this violates and explain why the original message ordering prevents this attack.

2. (8 points) **Non-Interactive Proofs with Fiat-Shamir**

   After applying the Fiat-Shamir transformation to the Schnorr identification protocol, Alice uses the resulting non-interactive proof to authenticate herself to a server. Later, the server publishes this proof transcript $(R, c, s)$ along with Alice's public key $Y$ as "proof that Alice authenticated at time $T$."

   (a) Explain why this published transcript constitutes undeniable evidence that Alice (specifically, someone with knowledge of the secret $x$) created it. What property of the Fiat-Shamir transformation makes forgery impossible?

   (b) Contrast this with the *interactive* Schnorr protocol. If a server recorded an interactive transcript $(R, c, s)$, explain why this is *not* convincing evidence to a third party. Specifically, describe how a malicious verifier could fabricate a valid-looking transcript without ever interacting with Alice.

## Answer Pages — Topic 2.8

## Answer Pages — Topic 2.8

## Bonus Question

1. (10 points (bonus)) **OPAQUE, OPRFs, and Hardware Security Modules**

   (a) An Oblivious Pseudorandom Function (OPRF) allows a client to evaluate $F_k(x)$ where the server holds key $k$ and the client holds input $x$, such that the server learns nothing about $x$ and the client learns nothing about $k$ beyond $F_k(x)$.

   Consider a naive "password-hardening" scheme where the client sends $H(\text{password})$ to the server, which returns $F_k(H(\text{password}))$ as the hardened password. Explain why this is *not* an OPRF and describe the specific information leaked to the server. Then explain how the OPAQUE protocol uses a proper OPRF construction (based on blinding) to achieve password-authenticated key exchange where: (1) the server never sees the password or any value derived solely from it, and (2) an attacker who compromises the server's database cannot mount an offline dictionary attack without also knowing the server's OPRF key $k$.

   (b) A company deploys Hardware Security Modules (HSMs) to protect their TLS private keys. A junior engineer argues: "Since the HSM performs all signing operations internally and the key never leaves the device, we are completely protected against key theft even if our servers are fully compromised."

   Identify two distinct limitations of this HSM deployment that the engineer has overlooked. For each limitation: (1) describe a concrete attack scenario that remains possible despite the HSM, and (2) explain what additional security measures or HSM features would mitigate this specific threat. Your answer should address both real-time abuse of signing capabilities and long-term security considerations.

## Answer Pages — Bonus Question

## Answer Pages — Bonus Question