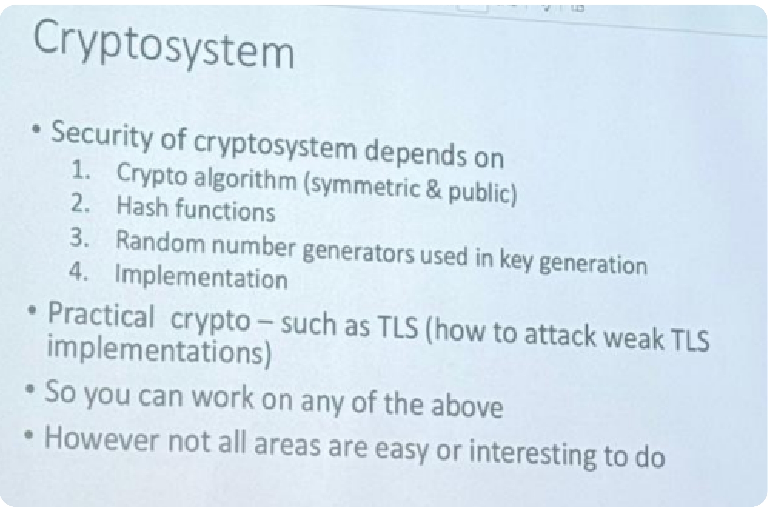
ChatGPT

✦ Get Plus ×

This is a copy of a conversation between ChatGPT & Anonymous.
Report conversation



Cryptosystem

- Security of cryptosystem depends on
  1. Crypto algorithm (symmetric & public)
  2. Hash functions
  3. Random number generators used in key generation
  4. Implementation
- Practical crypto – such as TLS (how to attack weak TLS implementations)
- So you can work on any of the above
- However not all areas are easy or interesting to do

suggest cryptography projects based on these topics (specific topics)

Based on the topics shown in your slide, here are **specific, concrete cryptography project ideas** you can actually implement or analyze. I've grouped them exactly according to the listed areas and kept them practical (good for coursework, final-year projects, or reports).

---

1. Cryptographic Algorithms (Symmetric & Public Key)

Symmetric Cryptography

- Implement **AES** and analyze how security breaks when:
  - ECB mode is used instead of CBC/GCM
  - Weak or repeated IVs are used
- Performance comparison of **AES vs ChaCha20** on different file sizes
- Demonstrate a **padding oracle attack** on CBC mode (simulation)

Public-Key Cryptography

- Implement **RSA** and show attacks caused by:
  - Small public exponent (e = 3)
  - Reusing primes
- Compare **RSA vs ECC** (key size, speed, security level)

- Implement **Diffie–Hellman key exchange** and demonstrate a **man-in-the-middle attack** when authentication is missing

---

2. Hash Functions

- Compare **SHA-1, SHA-256, SHA-3** in terms of:
  - Collision resistance
  - Speed
- Implement a **length extension attack** on SHA-256 (educational simulation)
- Demonstrate why **MD5 is broken** using real collision examples
- Build a **Merkle Tree** and show how it enables efficient data verification (blockchain-style)
- Password hashing project:
  - Compare **SHA-256 vs bcrypt vs Argon2**
  - Show why simple hashing is insecure

---

3. Random Number Generators (RNGs)

- Analyze weaknesses of **pseudo-random number generators** used incorrectly
- Implement a **secure vs insecure key generator** and show:
  - Predictable keys when RNG is weak
- Study real-world failures:
  - Debian OpenSSL RNG bug (case study + explanation)
- Compare:
  - `rand() vs /dev/urandom` vs cryptographic PRNGs
- Build a **randomness testing tool** (frequency, entropy tests)

---

4. Implementation Security

- Side-channel attack simulation:
  - Timing attack on password comparison
- Memory leakage analysis in crypto code
- Demonstrate how:
  - Hard-coded keys
  - Poor key storage

    break security

- Compare **secure vs insecure TLS certificate validation** in code
- Analyze common crypto implementation mistakes in open-source projects

---

5. Practical Cryptography (TLS Focus)

- Analyze **weak TLS configurations**, such as:
  - TLS 1.0 / 1.1 usage
  - Weak cipher suites
- Perform a **TLS scan** of real websites (using tools like SSL Labs) and explain vulnerabilities
- Simulate a **downgrade attack** on TLS
- Study historical attacks:
  - POODLE
  - BEAST
  - Heartbleed
- Build a **secure TLS server** and then intentionally weaken it to show attack impact

---

6. Applied / Integrated Projects (High-Impact)

- **Blockchain cryptography analysis**:
  - Role of hash functions, digital signatures, Merkle trees
- **Secure messaging system**:
  - End-to-end encryption using public + symmetric crypto
- **Password manager**:
  - Uses hashing, key derivation, encryption
- **Digital signature system**:
  - Compare RSA vs ECDSA
- **Secure file storage system**:
  - Encryption + key management + integrity checks

---

7. Research-Oriented / Theory-Light Projects

- Why cryptography fails in practice (case studies)
- Human factors in cryptographic security
- Regulatory challenges in deploying strong cryptography
- Comparison of crypto standards (NIST vs real-world usage)