CODE:

```
#include <iostream>
#include <string>
#include <openssl/sha.h>
#include <openssl/aes.h>
#include <openssl/rand.h>
using namespace std;

class SecuritySystem {
private:
   static const int AES_KEY_SIZE = 256;
   static const int BLOCK_SIZE = 16;
   string generateRandomKey() {
      string key;
      key.resize(AES_KEY_SIZE / 8);
      RAND_bytes(reinterpret_cast<unsigned char*>(&key[0]), AES_KEY_SIZE / 8);
      return key;
   }
   string encrypt(const string& plaintext, const string& key) {
      AES_KEY aesKey;
      if (AES_set_encrypt_key(reinterpret_cast<const unsigned char*>(&key[0]), AES_KEY_SIZE,
&aesKey) != 0) {
         cerr << "Error setting up AES encryption key." << endl;
         return "";
      }
```

```
        string ciphertext;
        ciphertext.resize(plaintext.size() + BLOCK_SIZE);
        AES_encrypt(reinterpret_cast<const unsigned char*>(&plaintext[0]),
                reinterpret_cast<unsigned char*>(&ciphertext[0]), &aesKey);
        return ciphertext;
    }

    string hash(const string& data) {
        unsigned char hash[SHA256_DIGEST_LENGTH];
        SHA256_CTX sha256;
        SHA256_Init(&sha256);
        SHA256_Update(&sha256, data.c_str(), data.length());
        SHA256_Final(hash, &sha256);
        string hashedData;
        for (int i = 0; i < SHA256_DIGEST_LENGTH; ++i) {
            char hex[3];
            snprintf(hex, sizeof(hex), "%02x", hash[i]);
            hashedData += hex;
        }
        return hashedData;
    }

public:
    // Perform confidentiality, integrity, and availability checks
```

```
    void secureData(const string& data) {
        // Step 1: Generate a random key for confidentiality
        string encryptionKey = generateRandomKey();

        // Step 2: Encrypt the data
        string encryptedData = encrypt(data, encryptionKey);
        if (encryptedData.empty()) {
            cerr << "Error encrypting data." << endl;
            return;
        }

        // Step 3: Calculate the hash for integrity
        string dataHash = hash(data);

        // Step 4: Simulate availability by printing the results
        cout << "Original Data: " << data << endl;
        cout << "Encrypted Data: " << encryptedData << endl;
        cout << "Data Hash: " << dataHash << endl;
    }
};

int main() {
    SecuritySystem securitySystem;
    string data = "Confidential Data";
```

```
    securitySystem.secureData(data);

    return 0;

}
```

OUTPUT:

Original Data: Confidential Data

Encrypted Data: bътud┬╣!цl^^|x╗ц╜дG

Data Hash: 5757d9a6e8768958eb1ee6864a5a2d361eac8b7d389c2e0944e56d161c7f727c