Problem 1: Suppose that to achieve resilience against data-center failures, a tenant would like to distribute a static file $F$ across three different data centers of a cloud provider, DC A, DC B, and DC C.

- Suppose that the tenant respectively stores $F_A$, $F_B$, and $F_C$ with DC A, DC B, and DC C. How can $F_A$, $F_B$, and $F_C$ be constructed such that: (1) If one data center fails, the tenant can still recover its file and (2) For any $F$, the total storage $|F_A| + |F_B| + |F_C|$ is substantially less than $3|F|$?

- DC A, DC B, and DC C sit on different continents (North America, Europe, and Asia respectively). If the tenant does not trust the cloud, sketch a remote testing protocol that enables the tenant to select a block of $F_X$ at random and challenge DC $X$ to prove that it is locally storing that block? What does the tenant need to know about its network connection with the DC to ensure a sound protocol? (You may assume that the tenant has a small amount (constant in $|F|$) of trustworthy local storage and controls hosts worldwide.)

- It is to be expected that the network bandwidth between DCs is vastly greater than that between any DC and the tenant, and the network latency lower. What might go wrong in a remote test in which the tenant performs a number of remote tests of block presence sequentially with a given DC?

- Suppose that each DC has a Hardware Security Module (HSM) that performs asymmetric encryption / decryption using a public key known to the tenant. Can these HSMs be used to remedy the problem surfaced in the last question? Why or why not?
Problem 2:

- What is the difference between pseudonymity and anonymity?

- What is the most common way for websites to leak users personal identifiers such as usernames and email addresses to third-party trackers, thus enabling the latter to identify pseudonymous profiles? Explain exactly how the leakage occurs.

- If you are a third-party tracker whose iframe is included into multiple websites, does canvas fingerprinting eliminate the need for third-party cookies? Explain.

Problem 3: libsafe is a wrapper around the C string library, intended to ensure that string operations cannot overwrite any control information stored on the stack (such as saved return address, saved frame pointer, etc.). For example, the libsafe wrapper around strcpy adds the following check before strcpy(src,dest) is executed:

\[ |\text{framePointer} - \text{dest}| > \text{strlen}(\text{src}) \]

- What additional protections are gained by using libsafe with canary-equipped executables?
• Give a short snippet of C code that contains a single call to a libsafe-protected strcpy, and yet is vulnerable to a memory corruption attack as a result of this call. Your attack must also bypass compiler-inserted stack canaries.

Problem 4: The sample exploit against Apache HTTPD described in the “Control Jujutsu” paper corrupts the data structure holding pointers to registered implementation functions of some Web server functionality and replaces a pointer to one of the registered functions with a pointer to a function chosen by the attacker (piped_log_spawn).

During normal execution, the pointer in question can never point to piped_log_spawn because the latter is not one of the registered functions. In other words, the control transfer executed during the attack is not actually part of the program’s control-flow graph (CFG).

Explain why this attack is nevertheless not prevented by fine-grained CFI. Give as much as detail as needed, i.e., don’t just say “pointer analysis is imprecise” but explain why the statically computed CFG permits the control transfer executed during the attack.