**Suggested Implementation: Cryptographic Analysis of Rolling Code Systems**

Here’s a step-by-step plan to implement a project based on this paper:

**1. Understand Rolling Code Systems**

* Research how Remote Keyless Entry (RKE) systems work, especially those using rolling codes.
* Familiarize yourself with the cryptographic concepts such as LFSRs (Linear Feedback Shift Registers) and ciphers like Hitag2 or AUT64.

**2. Simulate Rolling Code Communication**

* Create a simulation of rolling code transmission between a car key and the vehicle.
* Generate rolling codes using a simple encryption algorithm or an LFSR.
* Include a counter mechanism to simulate the incrementing nature of rolling codes.

**3. Implement Eavesdropping**

* Write a script that simulates capturing rolling code packets over a "radio channel."
* Store these captured packets for analysis.

**4. Cryptanalysis of Rolling Codes**

* Implement correlation-based attacks to predict future rolling codes by analyzing patterns in the intercepted packets.
* Focus on recovering the cryptographic key (e.g., Hitag2 key) using known methods described in the paper.

**5. Test the Attack**

* Simulate multiple attacks where your script eavesdrops a few rolling codes and attempts to unlock the simulated vehicle.
* Measure success rates and time required for key recovery.

**6. Build a Dashboard**

* Create a simple GUI or command-line interface showing:
  + Intercepted codes
  + Analysis of patterns
  + Predicted next code
  + Attack success status

**7. Propose Countermeasures**

* Simulate how improved cryptographic algorithms or protocols (e.g., rolling code validation with timestamps) could mitigate the attack.

**Tools and Technologies to Use**

1. **Programming Language**: Python (recommended for cryptographic simulations and signal processing).
2. **Libraries**:
   * NumPy/SciPy: For mathematical operations.
   * Cryptography: For simulating encryption algorithms.
3. **Optional GUI**: Use PyQt or Tkinter to visualize the system.