

Summer 2024 – CIS4951 Capstone Project

Multi-device & Cross-Platform 2-Factor Authentication TOTP Generator

Dillan Bailey (6364518)

- Generation of TOTP codes implementing all parts of the RFC6238 specification.
- State management to allow React.js components to reactively update to new information.
- Interaction with persistent data stores to preserve application data.
- Backend for multi-device synchronization.

```
export function useAccount(account: TOTPAccount) {
  const [state, setState] = useState<otp: string | null, timeLeft: number>([null, 0]);
  const otpRef = useRef<string | null>(null);

  const updateAccount = useCallback(async () => {
    const timeRemaining = Math.max(0, account.period - Math.floor(Date.now() / 1000 % account.period));

    if(otpRef.current === null || timeRemaining <= 1){
      otpRef.current = await generateOTP(account);
    }

    setState([otpRef.current, timeRemaining]);
  }, [account, setState]);

  useEffect(() => {
    updateAccount();

    const interval = setInterval(updateAccount, 1000);
    return () => clearInterval(interval);
  }, []);

  return state;
}

export async function generateOTP(account: TOTPAccount) {
  const counter = Math.floor(Date.now() / 1000 / account.period);

  //convert counter to bytes
  const dataView = new DataView(new ArrayBuffer(8));
  dataView.setBigUint64(0, BigInt(counter), true);

  const algorithmParams = {
    name: "HMAC",
    hash: WEBCRYPTO_ALGOS[account.algorithm]
  };

  const key = await crypto.subtle.importKey(
    "raw",
    decodeBase32(account.secret),
    algorithmParams,
    false,
    ["sign"]
  );

  const hmac = new Uint8Array(await crypto.subtle.sign(algorithmParams, key, dataView));
  const offset = hmac[hmac.byteLength - 1] & 0xF;

  const code = (
    ((hmac[offset] & 0x7F) << 24) ||
    ((hmac[offset + 1] & 0xFF) << 16) ||
    ((hmac[offset + 2] & 0xFF) << 8) ||
    (hmac[offset + 3] & 0xFF)
  ) % (10 ** account.digits);

  return code.toString().padStart(account.digits, "0");
}
```

