Balthazar Wallet: Making Password Authentication Practical on Web3 via OPAQUE and Privacy-Preserving Smart Contracts

Tomas Krajci Samuel Oleksak Ivan Homoliak

Brno University of Technology / Slovak University of Technology

Eurocrypt 2026

Motivation: The Problem

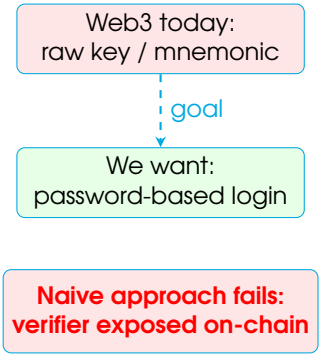
Web3 wallets have a UX problem:

- Users must manage raw private keys or mnemonics

The obvious fix - use passwords:

- Blockchain is public
- No server-enforced rate limits

Web2 passwords work because a trusted server enforces rate limits

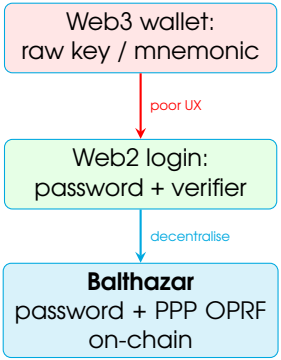


Motivation: Our Approach

Key insight:

- A Privacy-Preserving Platform (PPP) – a TEE-backed confidential EVM
- Contract state is encrypted so no observer (not even validators) can read stored secrets
- Every password guess requires an on-chain transaction: blockchain-enforced rate limiting

Result: Web2 password UX on a public, permissionless blockchain – without trusting any single server.



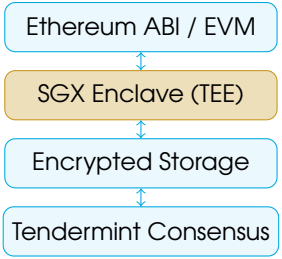
Background: TEE and Oasis Sapphire

Trusted Execution Environment (TEE):

- Isolated processor area
- Supports remote attestation: prove a specific binary runs inside a genuine enclave
- Intel SGX

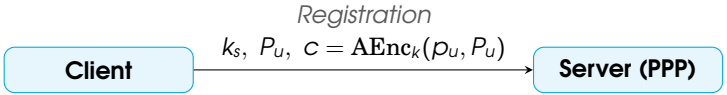
Oasis Sapphire (PPP instance):

- EVM-compatible; contracts run inside SGX enclaves
- Contract storage encrypted at rest with enclave-derived keys
- On-chain state is ciphertext – validators see nothing
- EVM precompile `0x05`: on-chain `modexp` for OPRF

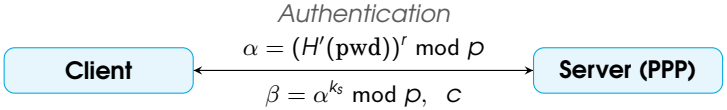


Key consequence for Balthazar:
The OPRF key k_s lives in TEE encrypted storage.

OPAQUE (Jarecki, Krawczyk, Xu – Eurocrypt 2018; RFC 9807): asymmetric PAKE with pre-computation resistance – server compromise does not enable offline dictionary attack.



Client generates k_s, derives $k{=}H((H'(\mathrm{pwd}))^{k_s})$, encrypts envelope c



Client unblinds: $k = H(\beta^{1/r} \bmod p)$, decrypts $c \to (p_u, P_u)$

Key property: k_s never leaves the server (PPP enclave).

Problem and Desired Properties

Goal: Secure, decentralised password-based wallet using OPAQUE on a PPP.

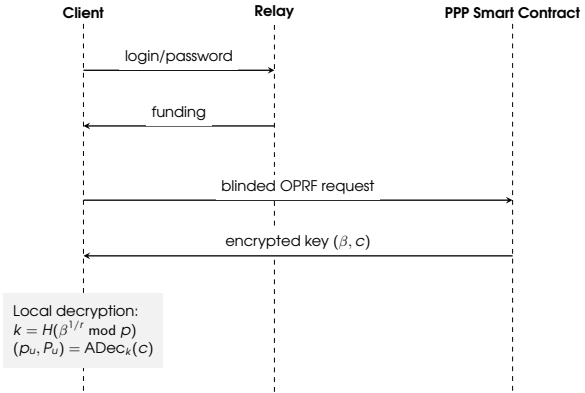
Desired Properties

- **Password confidentiality:** password and derived values never leave the client
- **Offline attack resistance:** every guess requires an on-chain OPRF call
- **Server compromise resilience:** enclave storage exposure does not enable offline guessing
- **Relay transparency:** relay learns no secrets, cannot impersonate users
- **Practical deployability:** standard Solidity tooling; minimal gas overhead

Threat Model

- **Network adversary:** reads all public chain state and transaction data
- **Malicious relay:** may attempt replay or registration squatting
- **Honest-but-curious operator:** cannot extract k_s from TEE encrypted storage
- **TEE assumption:** SGX provides confidential execution;

System Architecture



Client:

- Blinds password for OPRF
- Ephemeral keypair (x_u, X_u) per session
- Decrypts envelope locally

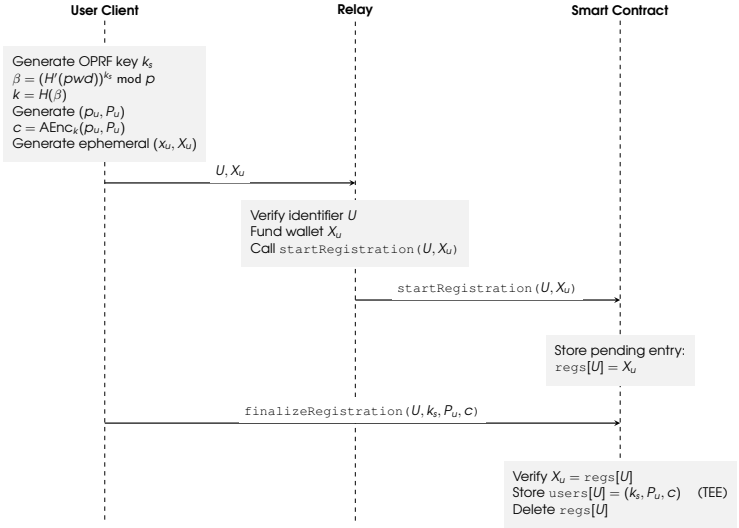
Relay:

- Verifies user identifier (email)
- Funds X_u – gasless onboarding
- Forwards txs to PPP contract

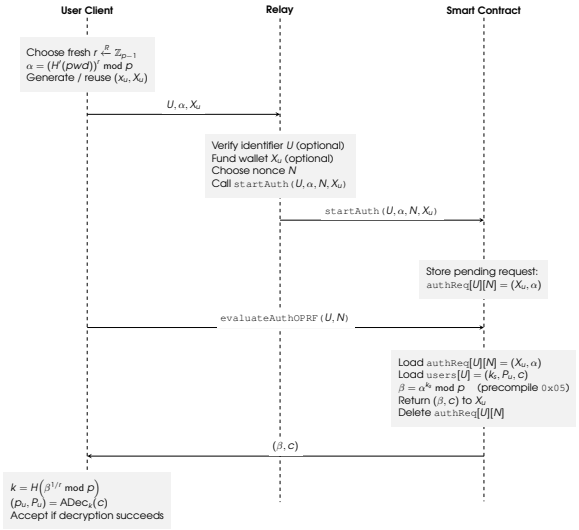
PPP Smart Contract (TEE):

- Stores $\texttt{users}[U] = (k_s, P_u, c)$ in TEE encrypted storage
- Evaluates OPRF: $\beta = \alpha^{k_s} \bmod p$
- k_s never leaves the enclave

Registration Protocol



Authentication Protocol



Security Analysis (1/2)

Password secrecy

- Password and unblinded OPRF output never leave the client
- Relay sees only $\alpha = (H'(\mathbf{pwd}))^r$

Offline attack resistance

- Every guess requires an on-chain transaction: blockchain-enforced rate limiting
- No password-derived verifier stored in clear

Server compromise resilience

- Attacker obtains (k_s, P_u, c) from TEE encrypted storage

Security Analysis (2/2)

Relay transparency / Registration squatting

- Relay is untrusted: sees only α (blinded), cannot learn k_s or envelope contents
- Two-step registration: `finalizeRegistration` verifies `msg.sender == regs[U]` – relay cannot substitute its own X_u

Replay protection

- **Auth replay:** fresh nonce N per session; `authReq[U][N]` deleted on first `evaluateAuthOPRF` – no second evaluation
- **TEE replay:** pending request stored on-chain, consumed atomically inside enclave
- **Sig replay:** ephemeral X_u per session; relay rejects previously seen client signatures

Blockchain confidentiality

- All contract state and execution traces are encrypted inside the TEE

Evaluation: Gas Costs and Latency

Experimental setup:

- Sapphire Localnet (SGX-backed, Oasis Protocol Foundation)
- Hardhat + TypeScript client + EVM precompile `0x05`
- Two OPRF prime sizes: 2048-bit and 1024-bit

Operation	2048b	1024b	Latency
`startRegistration`	111k	111k	$<$4023 ms
`finalizeRegistration`	164k	164k	$<$4102 ms
`startAuth`	338k	213k	$<$4059 ms
`evaluateAuthOPRF`	158k	87k	$<$4115 ms
Auth total	**496k**	**300k**	
Reg total		**≈275k**	

Key observations:

- Registration cost is independent of prime size - client evaluates OPRF locally, no on-chain modexp
- `startAuth` differs by prime size: stores α on-chain (1024 or 2048 bits)
- `evaluateAuthOPRF`: only on-chain modexp, $\beta = \alpha^{k_s} \bmod p$
- All operations $<$4.2 s: acceptable for a login flow

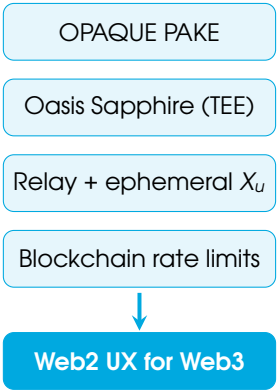
OPRF prime trade-off:

- 1024-bit: gas-efficient; NIST-deprecated (2013)
- 2048-bit: NIST-approved; ≈2× gas on modexp

Conclusion

Contributions:

1. First adaptation of OPAQUE to a public, EVM-compatible, TEE-backed smart contract platform
2. OPRF evaluation via EVM modexp precompile – pure Solidity, no off-chain cryptography server
3. Relay service for gasless, identifier-verified onboarding
4. Full prototype on Oasis Sapphire with gas and latency benchmarks: $\approx$496k / 300k gas per auth (2048b / 1024b)



Future Work

Account Recovery

2-of-3 threshold scheme:
password k_s TOTP secret recovery key

Any two factors suffice to recover the account.

Liquefaction Integration

Policy contracts gated by Balthazar auth level:

- Level 1 (password) – low-risk ops
- Level 2 (pwd + TOTP) – high-risk ops

Web2 UX for Liquefaction key encumbrance.

Thank You for Your Attention!

OPAQUE on TEE
Registration + Auth

Blockchain-enforced
rate limiting

Gasless onboarding
via relay + X_u

Tomas Krajci Samuel Oleksak Ivan Homoliak
Brno University of Technology / Slovak University of Technology
{ikrajci, ioleksak, homoliak}@fit.vut.cz

Backup: Notation

- $H(x)$ – cryptographic hash function
- $H'(x)$ – hash-to-integer function (OPRF input)
- $a^b \bmod p$ – modular exponentiation over prime p
- $\mathbf{AEnc}_k(m)$ / $\mathbf{ADec}_k(c)$ – AES-GCM authenticated enc/dec
- U – user identifier (verified email address)
- k_s – server-side OPRF private key (in TEE)
- $\beta = (H'(\mathbf{pwd}))^{k_s} \bmod p$ – unblinded OPRF output
- $k = H(\beta)$ – envelope encryption key
- (p_u, P_u) – user secret/public keypair (inside envelope)
- $c = \mathbf{AEnc}_k(p_u, P_u)$ – encrypted credential envelope
- (x_u, X_u) – ephemeral blockchain keypair (one per session)
- $\mathbf{regs}[U]$ – pending registration record (contract)
- $\mathbf{users}[U] = (k_s, P_u, c)$ – long-term user metadata (TEE)
- $\mathbf{authReq}[U][N] = (X_u, \alpha)$ – pending auth request (contract)

Backup: Future Work – TOTP and Sessions

On-chain TOTP second factor (planned):

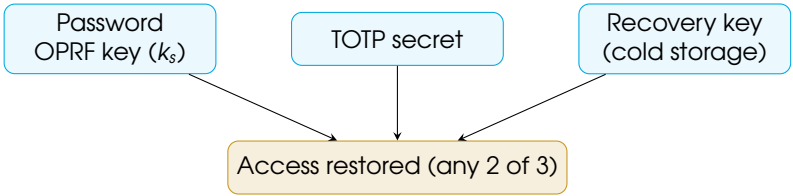
- HMAC-SHA-256, RFC 6238; 6-digit code, 30 s window, ± 1 drift
- TOTP secret derived inside TEE and stored in encrypted storage
- Replay guard: `lastOtpWindow` prevents code reuse
- Client calls `evaluateAuthOPRFWithOTP`$(U, N, \mathbf{code})$

Auth-level session model (planned):

- Level 1 (password only) for low-risk operations
- Level 2 (password + TOTP) for high-risk operations
- Sessions queryable by any contract via `isSessionValid(xu, requiredLevel)`
- Enables **Liquefaction integration**: policy contracts (key encumbrance, Dark DAOs, token renting) gated by Balthazar auth level – Web2 UX for TEE-backed credential sharing

Backup: Account Recovery Design

Planned 2-of-3 recovery scheme:



- Recovery operates on Balthazar (Sapphire) only; does not grant Liquefaction delegation authority directly
- All three factor secrets stored inside TEE encrypted storage
- Implementation: 2-of-3 threshold check or Shamir Secret Sharing
- Not yet implemented – account recovery is planned