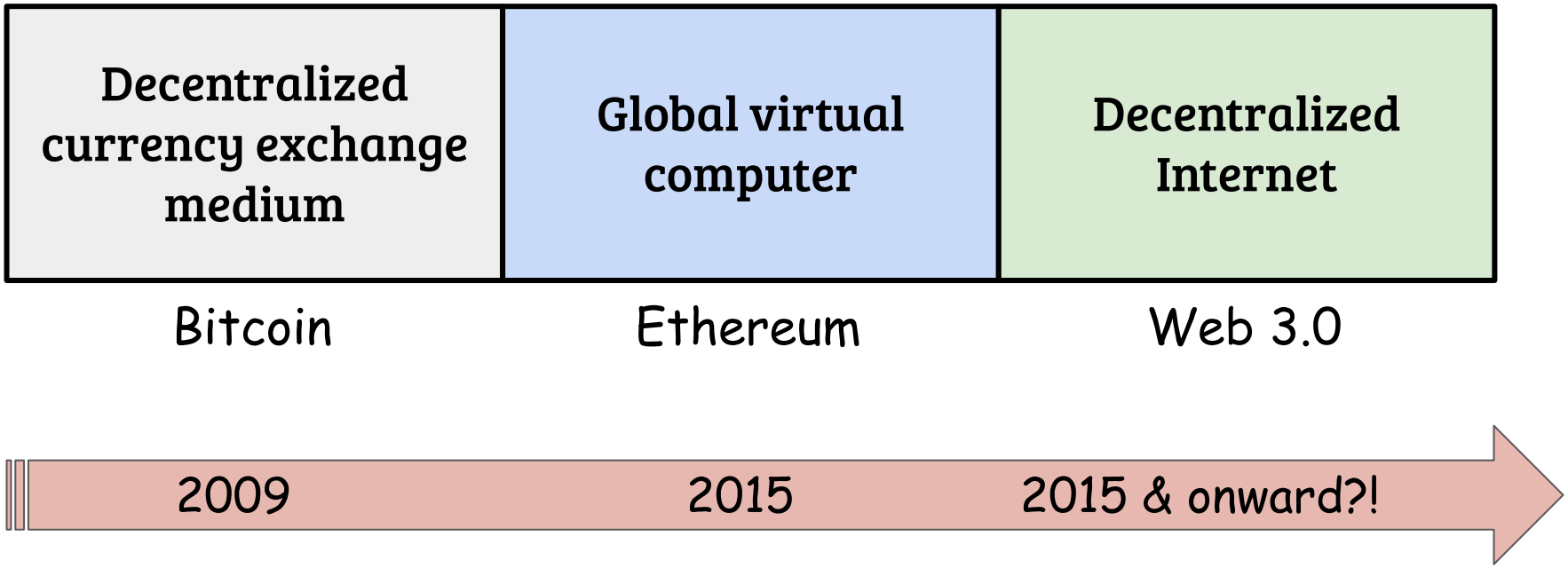
Anonymity in Decentralized Settings

Ghada Almashaqbeh

University of Connecticut

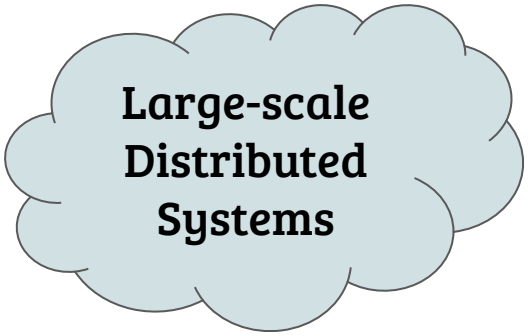
Anonymity Day, Columbia University
April 2025

The Decentralized Internet—Web 3.0



The Decentralized Internet—Web 3.0

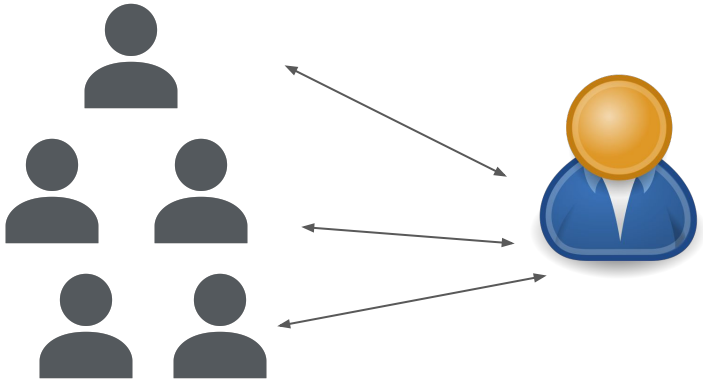
Decentralized currency exchange medium	Global virtual computer	Decentralized Internet
Bitcoin	Ethereum	Web 3.0



- Security
- Privacy
- Efficiency
- Viable applications
- …

Private Machine Learning

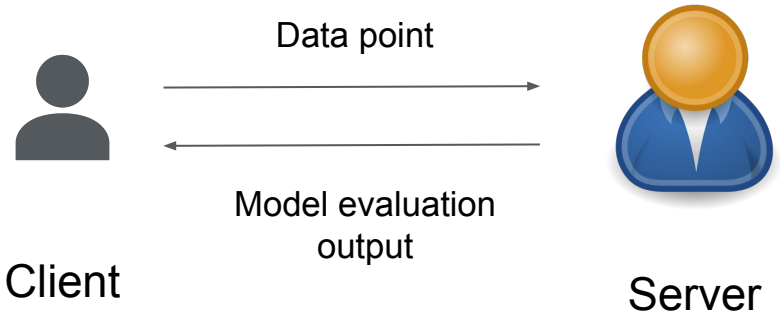
Federated Learning



Clients (users with private datasets)

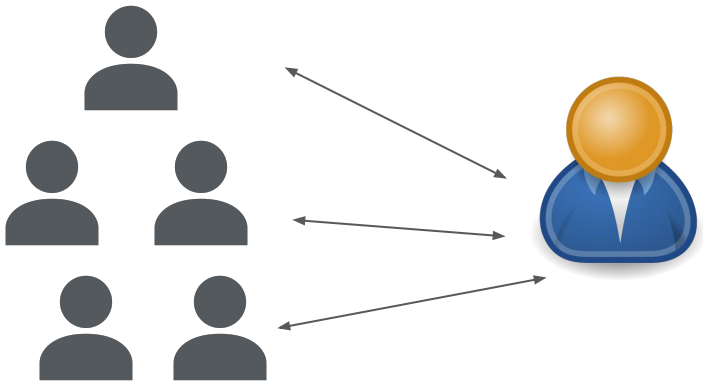
Server (or model owner)

Inference

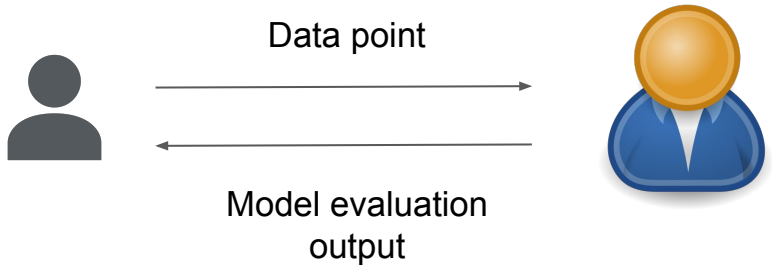


Private Machine Learning

Federated Learning



Inference



Also several security, privacy, and performane concerns!

This talk

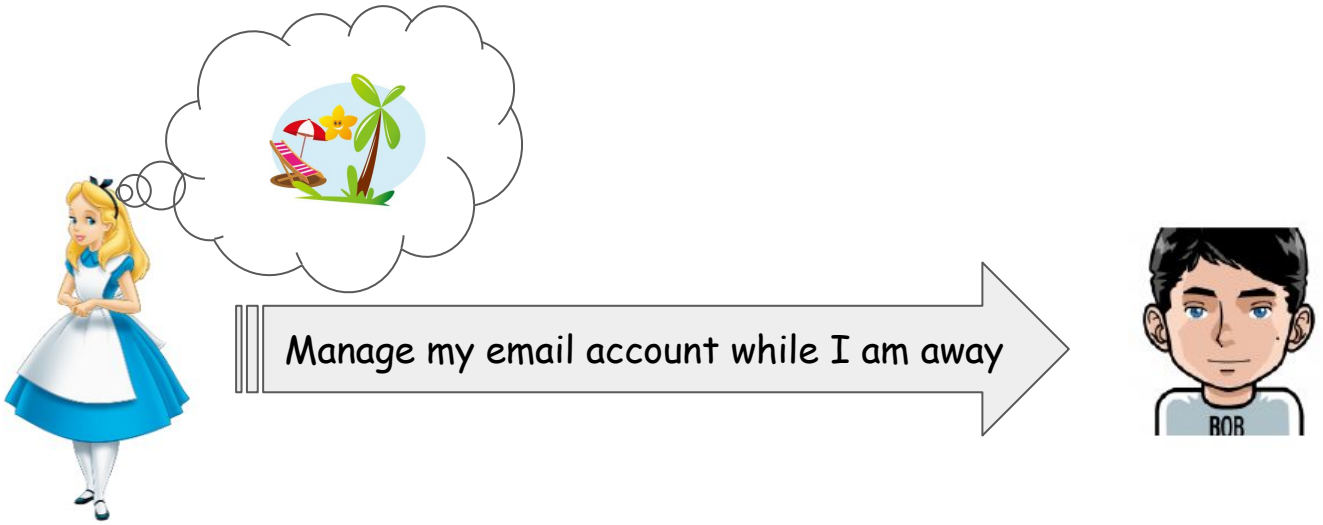
Anonymity in Two Decentralized Settings

- Anonymous signature delegation for Web 3.0
  - RelaySchnorr
- Anonymous participation in federated learning
  - AnoFel

RelaySchnorr: Anonymous, Timed and Revocable Proxy Signatures

*Ghada Almashaqbeh and Anca Nitulescu, *Anonymous, Timed and Revocable Proxy Signatures*, in ISC 2024 (https://eprint.iacr.org/2023/833).

Signature Delegation (Proxy Signatures)



Signature Delegation (Proxy Signatures)

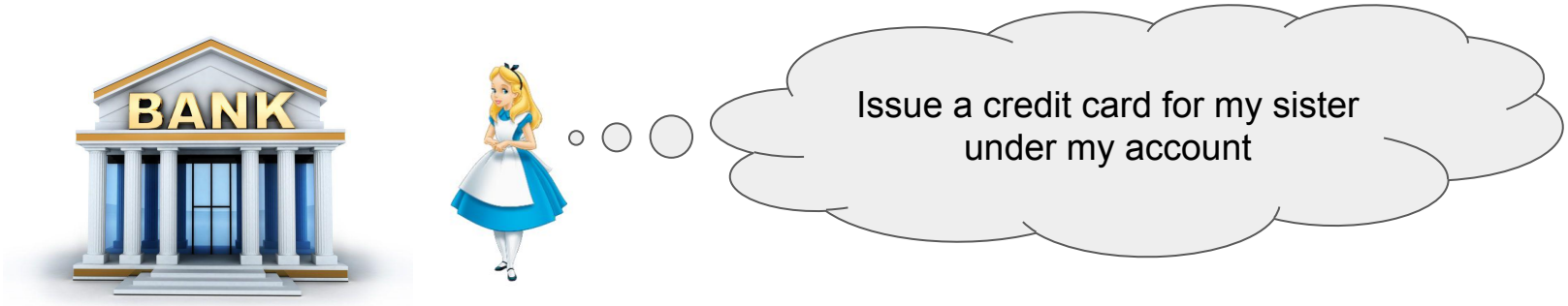
Manage my email account while I am away

Share the workload of handling emails

Produce signed messages on Alice's behalf

Motivating Applications

Can DeFi (decentralized finance) replace traditional banking services?

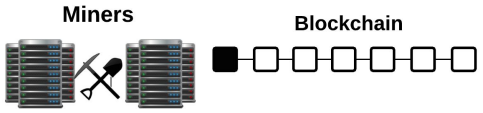


Motivating Applications

Can DeFi (decentralized finance) replace traditional banking services?

Issue a credit card for my sister under my account

Mitigating Targeted Attacks



I am on this consensus committee. If I do not sign within a timeout, sign on my behalf.

Desired Delegation Properties

- Anonymity of delegation.
- Timed delegation.
- Revocability.
- Policy enforcement.
- Decentralization.
- Non-interactivity.

Limitations of Prior Work

- No prior scheme achieved all these properties:
  - Many violate anonymity,
  - supported anonymity and policy enforcement without any revocation capability or timed notion,
  - or achieved revocability/timed notion at the expense of being interactive and/or involving a trusted third party.
- No formal security notion of proxy signatures encompassing all these properties.

This work

Can we do Better? … RelaySchnorr

- We define a security notion for anonymous, timed and revocable proxy signatures.
- We show a construction called **RelaySchnorr**
  - Combines Schnorr signatures, timelock encryption, and a public bulletin board.
  - Achieves all the desired properties listed before.
- We formally prove security of our scheme based on our notion.

Building Blocks - Schnorr Signatures

For a security parameter λ, let $\mathbb{G}$ be a cyclic group of a prime order q and a generator G, and $\mathsf{H} : \{0,1\}^* \times \mathbb{G}^2 \to \mathbb{Z}_q$ be a hash function. The Schnorr signature scheme is a tuple of three algorithms $\Sigma_{\mathsf{Schnorr}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ defined as follows:

- $\mathsf{Schnorr.KeyGen}(1^\lambda)$: On input the security parameter λ, choose uniform $x \in \mathbb{Z}_q$ and compute $X = G^x$. Set the secret signing key $\mathsf{sk} = x$ and the public verification key $\mathsf{vk} = X$.
- $\mathsf{Schnorr.Sign}(\mathsf{sk}, m)$: On input the secret key $\mathsf{sk} = x$ and the message m, choose uniform $k \in \mathbb{Z}_q$. Compute $K = G^k$, $X = G^x$, $c = \mathsf{H}(m, X, K)$, and $s = k + cx \mod q$. Output the signature $\sigma = (c, s)$.
- $\mathsf{Schnorr.Verify}(\mathsf{vk}, m, \sigma)$: On input the public key $\mathsf{vk} = X$, the message m, and signature $\sigma = (c, s)$ over m, compute $K = G^s \cdot X^{-c}$ and $c' = \mathsf{H}(m, X, K)$, then output 1 if $c = c'$.

Building Blocks - TLE

A Timelock encryption scheme $\mathcal{E}$ is a tuple of five PPT algorithms defined as follows:

TLE.Setup$(1^\lambda) \to (\mathsf{pp}, s)$: Takes as input the security parameter λ, and outputs public parameters pp and a private key s.

TLE.RoundBroadcast$(s, \rho) \to \pi_\rho$: Takes as input the round number ρ and a private key s, and outputs the round-related decryption information π_ρ.

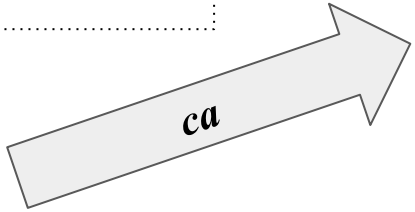
TLE.Enc$(\rho, m) \to (\mathsf{ct}_\rho, \tau)$: Takes as input the round number ρ and a message m, and outputs a round-encrypted ciphertext ct_ρ, and trapdoor τ for pre-opening.

TLE.Dec$(\pi_\rho, \mathsf{ct}_\rho) \to m'$: Takes as input the round-related decryption information π_ρ and a ciphertext ct_ρ, and outputs a message m'.

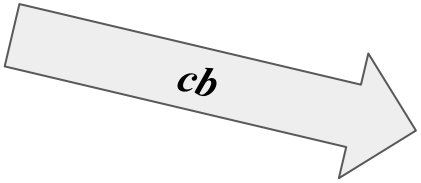
TLE.PreOpen$(\mathsf{ct}_\rho, \tau) \to m'$: Takes as input a ciphertext ct_ρ and a trapdoor τ, and outputs a message m'.

RelaySchnorr Construction

Delegation period [Ta, Tb]



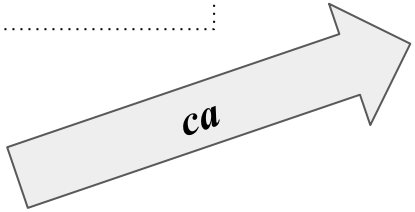
(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***



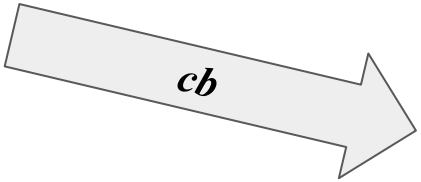
Bulletin board

RelaySchnorr Construction

Delegation period [Ta, Tb]



(1) Generate u random elements $k1, \ldots, ku$
(2) Generate u tokens: $t1, \ldots, tu$ (each token is a Schnorr signature over ki)
(3) Encrypt the tokens to time Ta, and the k values to time Tb

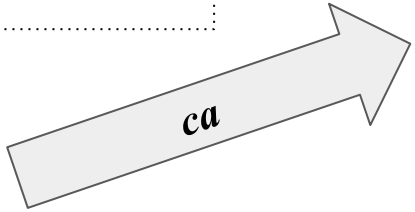


Bulletin board

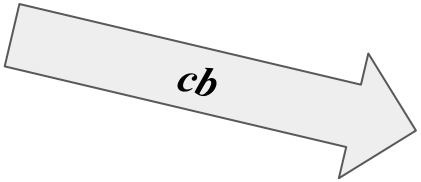
At time Ta:
(1) Decrypt the tokens.
(2) Use a token to sign message m (produce another Schnorr signature using the token).

RelaySchnorr Construction

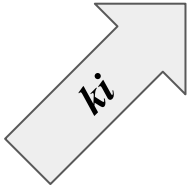
Delegation period [Ta, Tb]



(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***



Bulletin board



Verify a signature

At time ***Ta***:
(1) Decrypt the tokens.
(2) Use a token to sign message ***m*** (produce another Schnorr signature using the token).

RelaySchnorr Construction

One-time tokenizable Schnorr

Delegation period [Ta, Tb]

ca

(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***

cb

Bulletin board

ki

Verify a signature

At time ***Ta***:
(1) Decrypt the tokens.
(2) Use a token to sign message ***m*** (produce another Schnorr signature using the token).

RelaySchnorr Construction

Delegation period [Ta, Tb]

At time ***Tb***: Decrypt ***cb***

Publish all unused ***ki***

Bulletin board

RelaySchnorr Construction

Delegation period [Ta, Tb]

Board validators

At time ***Tb***: Decrypt ***cb***

Publish all unused ***ki***

Bulletin board

Before time ***Tb***

Publish all unused ***ki***

RelaySchnorr Construction

Delegation period [Ta, Tb]

Board validators

At time ***Tb***: Decrypt ***cb***

Publish all unused ***ki***

Bulletin board

Before time ***Tb***

Publish all unused ***ki***

Timed delegation

Automatic and on-demand revocation

Anonymity and Policy Enforcement

- **Anonymity is achieved because:**
  - Proxy signer identity is not included.
  - Delegation info is sent privately to the proxy signer.
  - The signature structure is the same for both the original or proxy signer, and verified using the same Verify algorithm.
  - Original signer mimics the behavior of having a delegation for her signatures.
- **Policy enforcement over messages:**
  - Conventional methods from the literature: public warrants and private ones (using NIZKs).

Issues in Practice

- Denial of service attacks against the signer.
- Bulletin board synchronization.
- Off-chain processing issues.
- Information lookup cost.

Security

Theorem 1. *Assuming EUF-CMA security of Schnorr signatures, the schnorr-koe assumption, a secure bulletin board, a CCA-secure TLE scheme, an EUF-CMA secure signature scheme, and a secure NIZK proof system, RelaySchnorr is an anonymous, timed and revocable proxy signature scheme (cf. Definition 2).*

- Unforgeability relies on the unforgeability of Schnorr signatures in the random oracle model, and the Schnorr knowledge of exponent assumption.
- Anonymity is achieved by having identical signature structure and behavior.
- Revocability relies on the security of timelock encryption and the bulletin board.
- Policy enforcement relies on the security of digital signatures (for public warrants) or NIZKs (for private policies), as well as security of timelock encryption and the bulletin board.

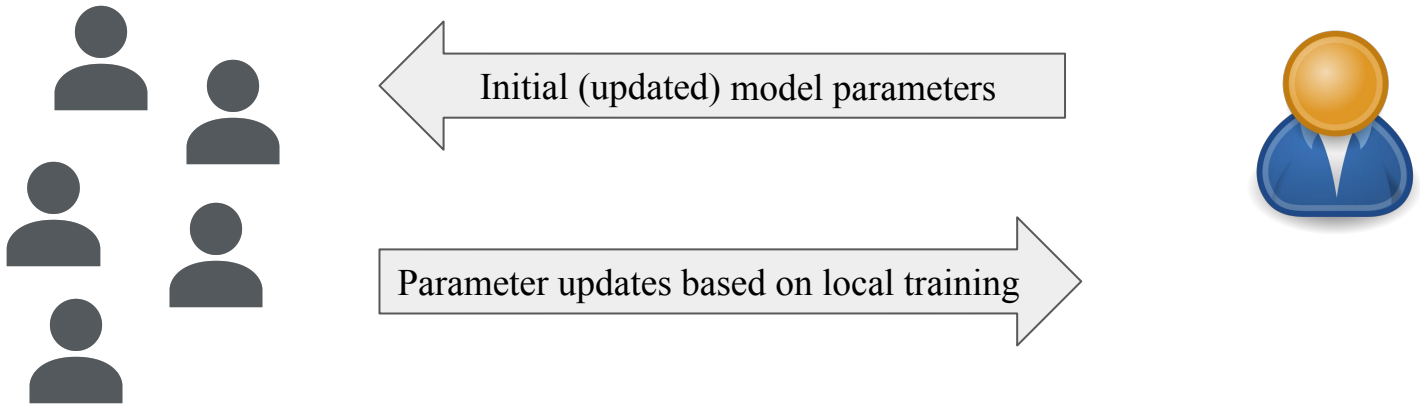
AnoFel: Supporting Anonymity for Privacy-Preserving Federated Learning

*Ghada Almashaqbeh and Zahra Ghodsi, *AnoFel: Supporting Anonymity for Privacy-Preserving Federated Learning*, in PETS 2025 (https://petsymposium.org/popets/2025/popets-2025-0051.pdf).

Overview - Federated Learning

Users with private datasets

Server with ML model to train

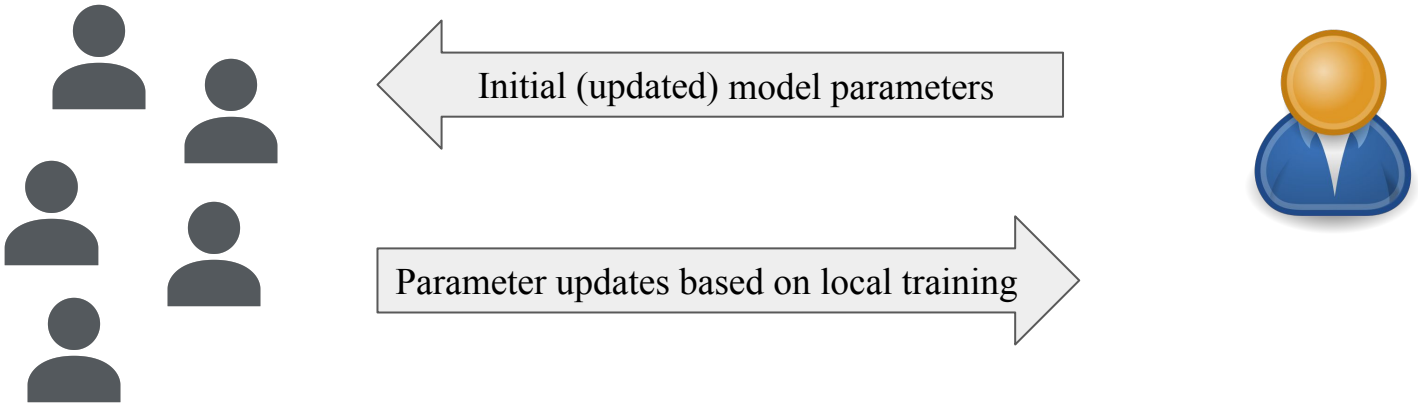


- Training is divided into rounds or iterations.
- Each iteration starts with receiving initial model parameters from the server.
- Each client trains the model locally and then submit updates (gradients) to the server.
- Server aggregates these submissions and updates the model accordingly, then shares the updated model with the clients to start a new training iteration.
- Continues until the model converges.

Overview - Federated Learning

Users with private datasets

Server with ML model to train



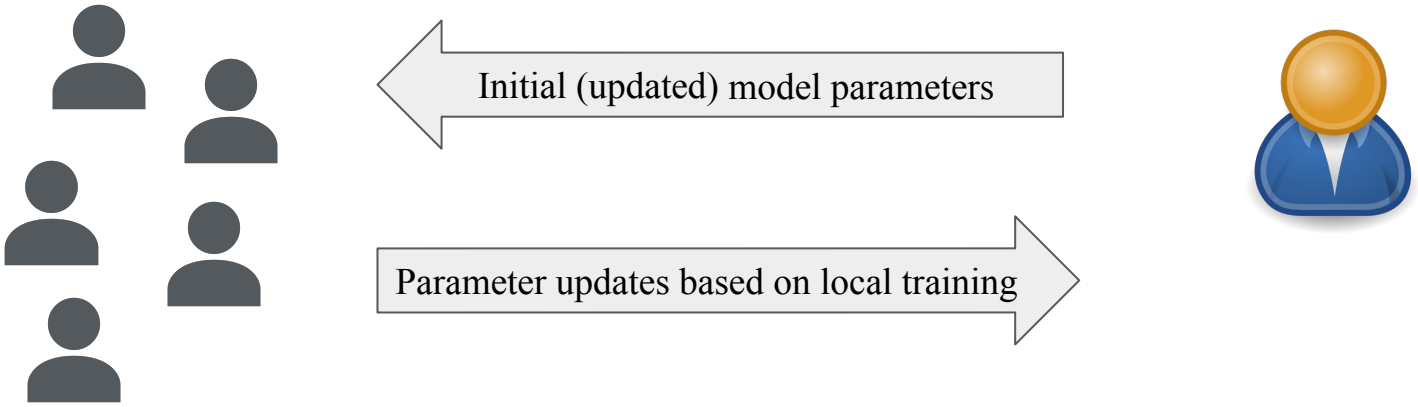
Several design, security and privacy concerns:

- Individual model updates may leak information ⇒ secure aggregation.
- Poisoning attacks ⇒ ensure that only legitimate datasets are used.
- Inference and membership attacks ⇒ differential privacy
- Dynamic participation.
- And …

Anonymous Participation is Critical

Users with private datasets

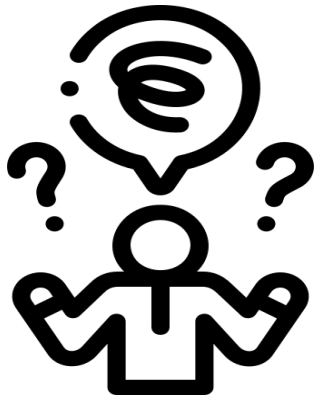
Server with ML model to train



- Usually training happens among a set of clients known to the server (and they communicate directly).
- The mere knowledge of participation may evade privacy.
  - Training over sensitive data related to user health, ethnicity, etc.
  - Being a participant implies that a client, e.g., suffers from a particular disease or belongs to a particular group population.

Can we build a private federated learning framework that

1. supports dataset privacy,
2. allows training in static and dynamic settings, and
3. achieves client participation anonymity?

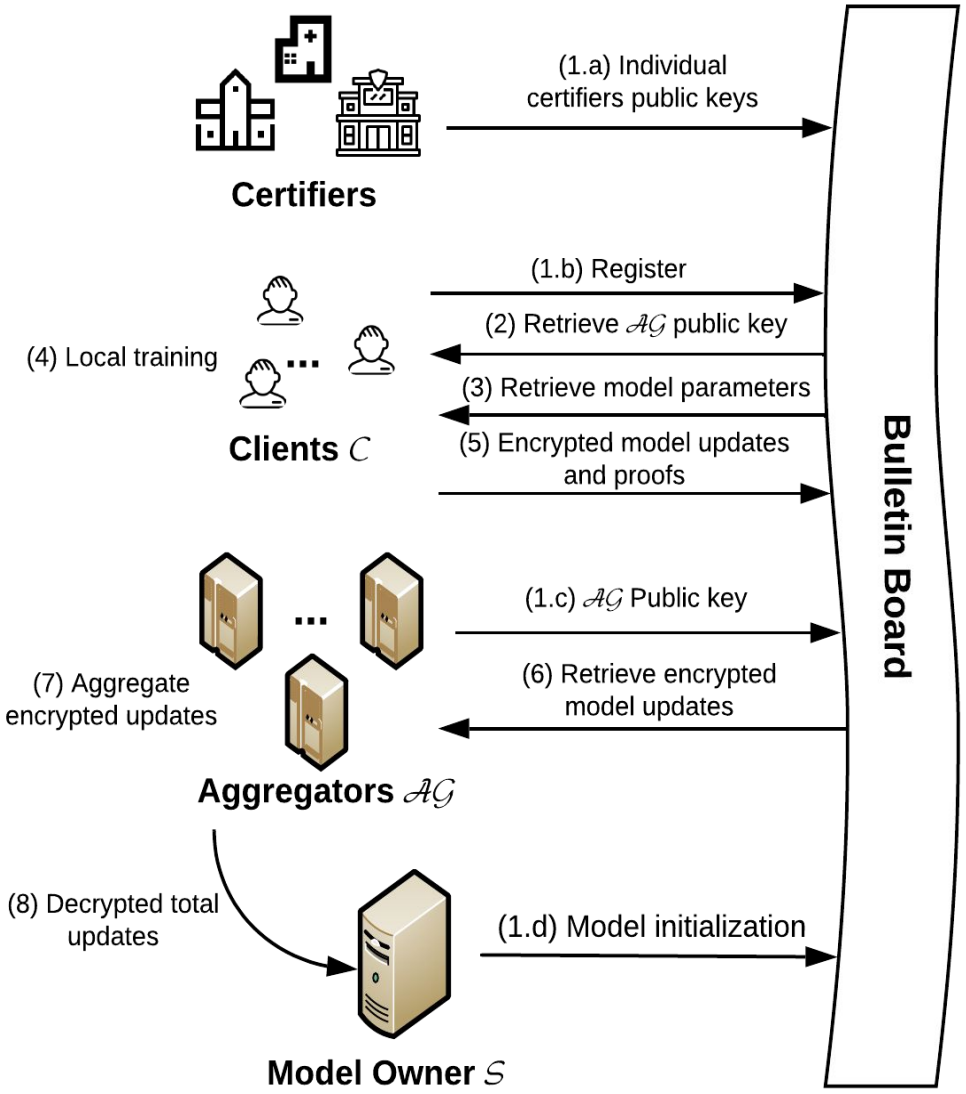


The AnoFel Framework

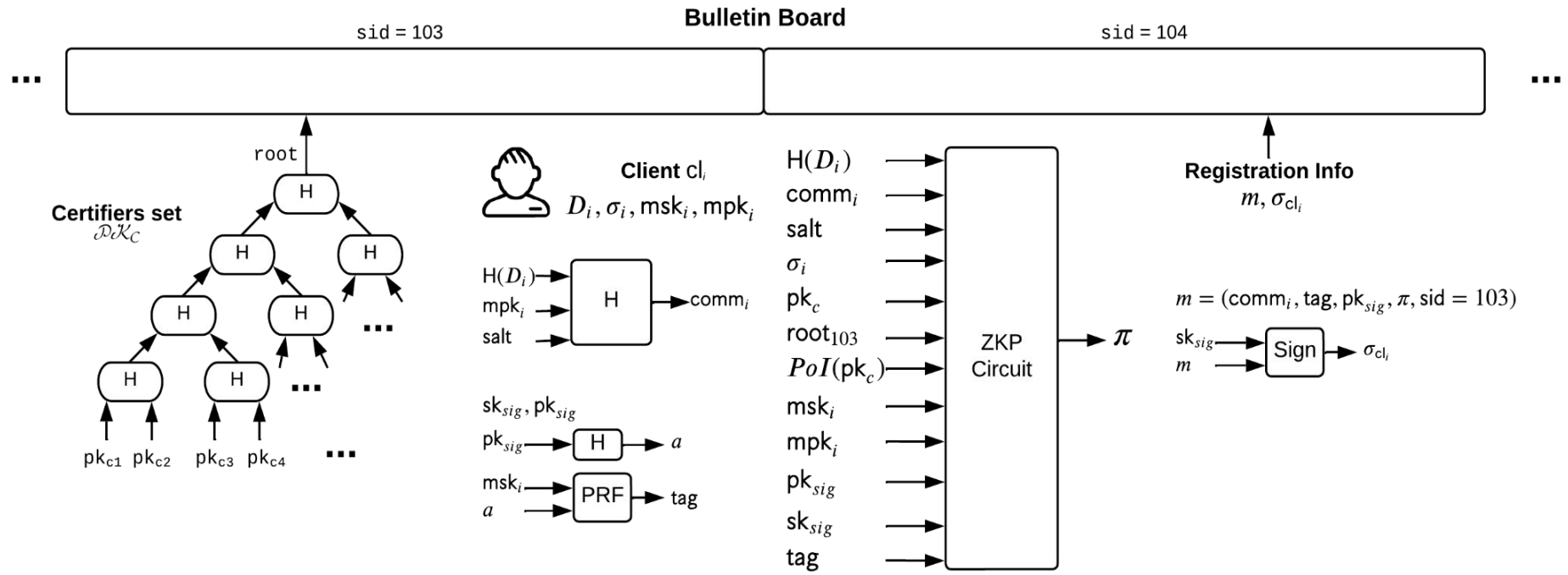
- Leverages several cryptographic primitives (threshold homomorphic encryption, commitments, and non-interactive zero knowledge proofs), the concept of anonymity sets, differential privacy, and a public bulletin board in its design.
- Supports anonymous user registration, as well as unlinkable and confidential model update submission.
- Allows dynamic participation without needing any recovery protocol or interaction with the server/other clients.

AnoFel Design

- A set of certifiers, clients, aggregators, and a server.
- Server is malicious.
- Aggregators are semi-honest and <= t-1 can be corrupt.
- Clients are malicious during registration but semi-honest during training.
- Works in the random oracle model.
- *Anonymity at the network layer level is out of scope.*

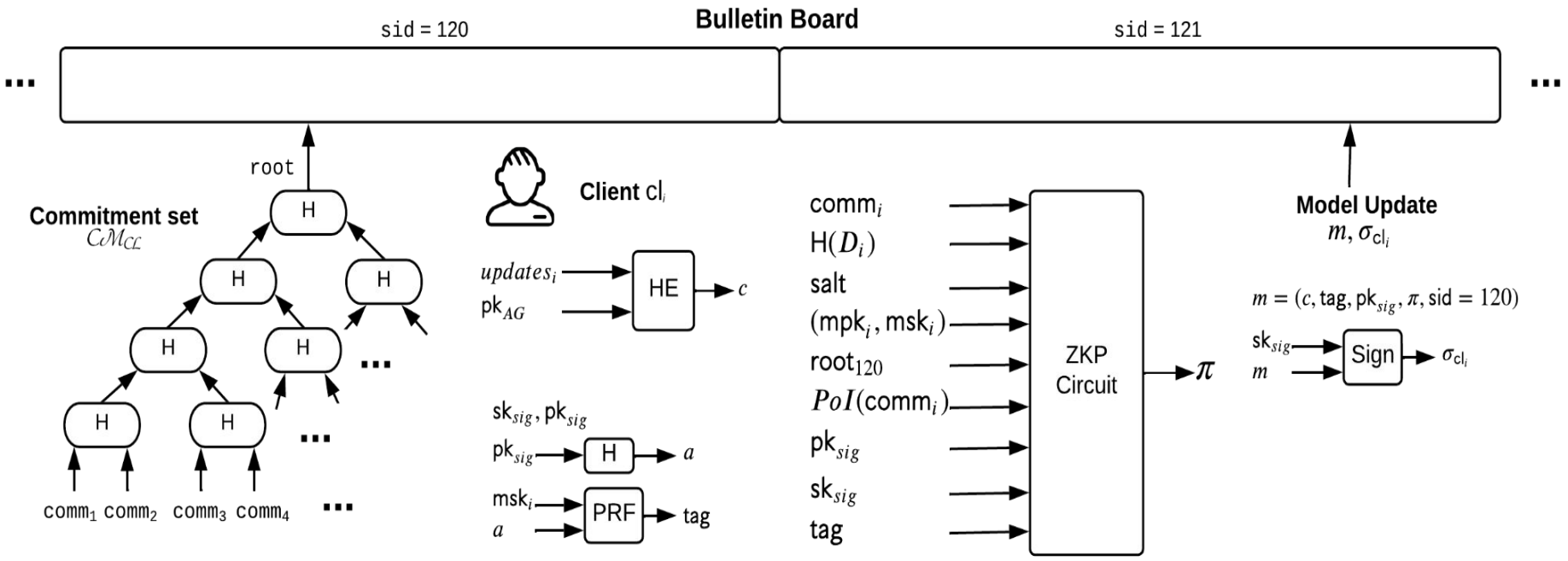


Client Registration Phase



- Owns a certified (signed) dataset.
- Post a commitment to the dataset and the master public key on the board.
- This is accompanied with a ZKP attesting that client owns a certified dataset but without revealing the certifier’s or the client's ID.
    - This ZKP is generated with respect to a particular board state.

Training Phase



- Encrypt the model updates under AG's public key.
- Generate a ZKP attesting that the client is a registered one and indeed knows the opening of a registered dataset.
  - As before, this ZKP is with respect to particular board state.

Model Access Phase

- The aggregator committee will aggregate the submitted updates.
  - Using homomorphic add.
- DP is applied at the aggregator level instead of the client level!
  - Each member in AG generates noise, using DP.noise divided by *t*, encrypt it and post the ciphertext on the board.
  - Each member collects the *n* noise ciphertexts.
  - Add these to the aggregated updates.
- Each member use their share of the decryption key to produce a partial decryption of the aggregated updates.
- The server, upon receiving *t* parial ciphertexts, can access the aggregated updates.

Supporting Dynamic Participation

- Each client can register at any time and without interacting with anyone.
- Same for training, no interaction with others is needed.
- If a client decides to drop from an iteration, it simply does not submit any updates.
  - This will not impact others nor the protocol flow.
  - AG is responsible for aggregation and decryption.
  - Communication is mediated by the board.

Security

- We define a notion for private and anonymous federated learning (PAFL).
  - Captures correctness, anonymity, and dataset privacy.
  - Parameterized by accuracy and privacy loss bounds to account for the use of non-cryptographic techniques.
- We formally prove that AnoFel satisfies this notion.

Performance Evaluation

- Focusing on the computation overhead imposed by the added techniques to achieve privacy/anonymity.
- Extensive benchmarks for various model sizes, datasets, and number of clients.
- We compare with two prior works (Bonawitz et al. [16], and Truex et al. [73])---none of them support anonymity and the former does not employ DP.
- We also evaluate accuracy for a range of number of clients and different dataset distributions.
- Highlights of the results:
  - Client setup < 3 sec and training is < 8 sec (AnoFel overhead is around 45% of training).
  - Up to 5x faster than Truex et al., and up to 2x slower than Bonawitz et al.

Last Stop!

Conclusion and Future Work

- Emerging technologies continue to introduce challenging settings for privacy and anonymity.
- Future work directions:
  - Explore delegation for other cryptographic primitives.
    - E.g., zero knowledge proofs (aka delegation of private wallets).
  - Achieve anonymity for federated learning without relying on a bulletin board and/or under stronger adversarial models.
  - Integrate solutions for network layer anonymity.

Thank you!

Questions?

ghada@uconn.edu
https://ghadaalmashaqbeh.github.io/

Concrete Construction

Setup phase

Let λ be a security parameter, S be the original signer, P be the proxy signer, and TLE be a timelock encryption scheme. Construct an anonymous, timed and revocable proxy signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Delegate}, \mathsf{DegSign}, \mathsf{Revoke}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, set $\mathbb{G}$ to be a cyclic group of a prime order q with a generator $G \in \mathbb{G}$ and $\mathsf{H} : \{0,1\}^* \times \mathbb{G}^2 \to \mathbb{Z}_q$ to be a hash function, initialize state $= \{\}$, and invoke $\mathsf{TLE.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{TLE.pp}, \mathsf{H}, \mathbb{G}, G, q, \mathsf{state})$.

$\mathsf{KeyGen}(1^\lambda)$: On input the security parameter λ, choose uniform $x \in \mathbb{Z}_q$, then compute $X = G^x$. Output the signing key $\mathsf{sk} = x$ and the verification key $\mathsf{vk} = X$.

Concrete Construction

Sign—used by original signer S

Sign(sk, m): On input the signing key sk $= x$ and some message m, do:

- Choose uniform $k, r, e \in \mathbb{Z}_q$, compute $R = G^r$, $E = G^e$
- Compute $w = \mathsf{H}(k, X, R)$, $z = (r + wx) \mod q$, and $Z = G^z$
- Compute $c = \mathsf{H}(m, Z, E)$ and $s = (e + cz) \mod q$ (if $z = 0$ or $s = 0$ start again with fresh r and e)
- Output the signature $\sigma = (w, c, s, k, Z)$

Every now and then, S either (1) populates a set klist from the stored k values and fresh ones, encrypts it as $(\mathsf{ct}_b, \tau_b) = \mathsf{TLE.Enc}(\mathsf{klist}, \rho_b)$, where ρ_b is some future round number, and posts (ρ_b, ct_b) on the board (resulting in state$'$[vk] $=$ state[vk] $\|$ (ρ_b, ct_b)), or (2) posts a fresh klist on the board (resulting in state$'$[vk] $=$ state[vk] $\|$ klist).

Concrete Construction

Delegate—invoked by original signer S

Delegate(sk, vk, degspec): On input the keypair ($\mathsf{sk} = x, \mathsf{vk} = X$) and delegation specifications $\mathsf{degspec} = (u, [\rho_a, \rho_b])$, where $u \in \mathbb{N}$ and $[\rho_a, \rho_b]$ is the delegation period, do the following:

- Set $\mathsf{klist} = \{\}$
- Do the following for $i \in \{1, \ldots, u\}$:
  - Choose uniform $k_i, r_i \in \mathbb{Z}_q$
  - Compute $R_i = G^{r_i}$ and $w_i = \mathsf{H}(k_i, X, R_i)$
  - Compute $z_i = (r_i + w_i x) \mod q$ (if $z_i = 0$ start again with fresh r_i)
  - Set $t_i = (z_i, w_i, k_i)$ and $\mathsf{klist} = \mathsf{klist} \cup \{k_i\}$
- Compute two ciphertexts: $(\mathsf{ct}_a, \tau_a) = \mathsf{TLE.Enc}(t_1 \| \cdots \| t_u, \rho_a)$ and $(\mathsf{ct}_b, \tau_b) = \mathsf{TLE.Enc}(\mathsf{klist}, \rho_b)$ (where τ_b is the revocation key rk).
- Set $\mathsf{degInfo} = (\rho_a, \rho_b, \mathsf{ct}_a)$
- Output $(\mathsf{degInfo}, \mathsf{ct}_b \| \tau_b)$

S stores ciphertext ct_b and trapdoor τ_b to be used for revocation if needed (τ_a is dropped as it is not needed), posts (ρ_b, ct_b) on the board (resulting in $\mathsf{state}'[\mathsf{vk}] = \mathsf{state}[\mathsf{vk}] \| (\rho_b, \mathsf{ct}_b)$), and sends degInfo to P.

Concrete Construction

Delegate Sign—used by proxy signer P

DegSign(m, degInfo): On input a message m and delegation information degInfo, P does the following (let ρ_{now} = state.round be the current round number):

- If $\rho_{now} < \rho_a$ or $\rho_{now} > \rho_b$, then do nothing
- If $\rho_a \leq \rho_{now} \leq \rho_b$, then:
  - If degInfo $= (\rho_a, \rho_b, \mathrm{ct}_a)$, then retrieve π_{ρ_a} from the board (π_{ρ_a} = state.roundInfo(ρ_a)) and set degInfo $= (\rho_a, \rho_b, \mathrm{TLE.Dec}(\pi_{\rho_a}, \mathrm{ct}_a))$
  - Pick an unused signing token $t = (z, w, k)$ from degInfo
  - Compute $Z = G^z$
  - Choose uniform $e \in \mathbb{Z}_q$ and compute $E = G^e$
  - Compute $c = \mathrm{H}(m, Z, E)$, and $s = e + cz \mod q$ (if $s = 0$ start again with a fresh e)
  - Output the signature $\sigma = (w, c, s, k, Z)$

Concrete Construction

Automatic/On demand revoke—invoked by validators or original signer S
Verify—Invoked by a verifier for any signature

Revoke(degInfo, rk, state[vk]): On input degInfo $= (\rho_b, \mathsf{ct}_b)$, revocation key rk, and revocation state state[vk], do (let $\rho_{now} =$ state.round be the current round number):

- If $\rho_{now} \geq \rho_b$, then retrieve π_{ρ_b} from the board ($\pi_{\rho_b} =$ state.roundInfo(ρ_b)) and compute klist $=$ TLE.Dec$(\pi_{\rho_b}, \mathsf{ct}_b)$
- If $\rho_{now} < \rho_b$, then use rk $= \tau_b$ to compute klist $=$ TLE.PreOpen(ct_b, τ_b)
- Add all k values such that $k \in$ klist $\wedge\, k \notin$ state[vk] to the board state state[vk] associated with vk resulting in an updated state state[vk]$'$.

Verify(vk, $m, \sigma = (w, c, s, k, Z)$, revState $=$ state[vk]): On input the verification key vk $= X$, the message m, signature $\sigma = (w, c, s, k, Z)$ over m, and the revocation state state[vk], if $k \in$ state[vk], then output 0. Else, add k to state[vk] (resulting in state$'$[vk] $=$ state[vk] $\|\, k$) and do the following:

- Compute $R = Z \cdot X^{-w}$ and $E = G^s \cdot Z^{-c}$
- Output 1 if and only if $w = \mathsf{H}(k, X, R) \;\wedge\; c = \mathsf{H}(m, Z, E)$.