Anonymous yet Verifiable Privacy-preserving Demand Response

Rosario Giustolisi[1], Emad Heydari Beni[2], Daniele Marletta[3], and Maryam Sheikhi Garjan[4]

[1] IT University of Copenhagen
[2] Nokia Bell Labs & KU Leuven
[3] University of Catania
[4] University of Hamburg

Abstract. Demand Response (DR) in energy systems is a flexibility mechanism enabling consumers to modify their electricity demand in response to signals from network operators, designed to ensure power grid reliability. In particular, incentive-based DR programs, in which consumers provide load reduction in exchange for financial remuneration, have proven more effective than alternative approaches such as price-based programs. However, incentive-based approaches have taken only partial account of privacy considerations, mainly because they require smart meters to disclose user energy baselines and consumption patterns to aggregators in order to determine rewards.
In this paper, we propose a privacy-preserving scheme that supports incentive-based DR programs while ensuring the confidentiality of user data and identities. We prove that our scheme provides data privacy, participation privacy, and public verifiability, and we present a prototype implementation together with a performance evaluation. Our results show that our construction is practical for real-world DR deployments with considerably large user populations.

1 Introduction

DR programs allow users to reduce their energy consumption in response to signals from energy operators who wants to cope with fluctuations in energy production, which can compromise the stability of electrical grids and disrupt operational planning. In particular, incentive-based DR programs, in which users receive financial remuneration in exchange for their reduction in energy consumption, have been shown to outperform other schemes such as price-based programs [28, 3, 26]. Despite their advantages, most incentive-based schemes only marginally address privacy concerns, since they require smart meters to disclose baseline data and detailed consumption profiles to aggregators in order to compute incentives. Moreover, user participation in DR programs has not been considered private information, increasing the risks of energy profiling. Such concerns represent one of the main barrier for a more widespread adoption of DR programs [12], and related work have confirmed that private information can be inferred from analysis of data from IoT devices such as smart meters [20, 4, 18].

In this work, we propose a privacy-preserving scheme that supports incentive-based DR programs while ensuring the privacy of user data and identities. Our scheme is based on a self-reported baseline methodology [19], which controls baseline inflation through random user selection and penalties for false reporting. We address the privacy challenges inherent in this approach, while enabling the most effective DR methodology to be deployed in Energy Communities.

One of the key ideas behind our scheme is the use of bitwise encryption to protect energy consumption values, combined with a threshold-comparison protocol that enables aggregators to perform privacy-preserving comparisons and ultimately compute a publicly verifiable reduction value.

Another key idea is the generation of anonymized public keys for registered smart meters, which, together with non-interactive zero-knowledge proofs (NIZKP) of knowledge of signature, ensures participation privacy. Most of the computation required to achieve participation privacy is verifiably outsourced to the aggregators, reducing both the computational and communication overhead for the smart meters. Moreover, our scheme preserves the privacy of individual consumption-reduction values as well as the final aggregated reduction value. Thus, our privacy guarantees span from individual data to aggregated outcomes and to the parties generating such data.

We prove that our scheme ensures data privacy and user anonymity if the underlying encryption scheme with the NIZKPs is IND-CCA1 secure and the signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA). Our scheme is the first publicly verifiable privacy-preserving DR scheme providing protection against either a malicious DR aggregator or a energy aggregator. Its complexity is linear in the number of participating users.

We provide a prototype implementation of the scheme in Python showing that our scheme's key primitives require only a few milliseconds to execute on a standard laptop. The end-to-end evaluation (scaling up to 1000 users) demonstrates that the total execution time of all phases grows linearly with the number of users. In the real world, the most time-critical phase of the protocol takes a couple of minutes. Our privacy-preserving results demonstrate that this phase can be completed within a few tens of seconds for a large number of participants.

2 Background

2.1 Self-Reported Baseline Mechanism (SRBM)

Incentive-based DR programs require users to disclose their energy consumption baselines in order to provide financial renumeration based on the energy reduction. SRBM [19] allows users who decide to participate in the DR program to self report their baselines. In particular, this mechanism is defined by the following key elements: (i) a low probability (e.g., 10%) for randomly selecting users to join the DR event and become eligible to receive the financial incentive; (ii) a reward function that specifies the payment awarded per unit of reduction provided by selected users; (iii) a penalty function applied to non-selected users

whose actual consumption during the DR event diverges from their baselines. Combining random user selection with penalties for untruthful baseline reports enables to control baseline inflation, which occurs when users manipulate the system by reporting inflated baseline values.

While SRBM addresses several limitations of traditional baseline estimation methods, data privacy remains a major concern because the mechanism requires users to reveal their baselines and metering data. Moreover, user identities are also disclosed, which increases the risk of energy profiling. Finally, the mechanism assumes that all entities are honest, and does not address the presence of malicious actors who may attempt to gain unauthorized access to data or tamper with information to disrupt the system.

2.2 Parties and Trust Model

An incentive-based DR program involves the following parties.

Distribution System Operator (DSO). The DSO assesses the grid status and defines an incentive-based DR program during periods of high grid stress, to incentivize end users to reduce consumption. The DSO defines the DR parameters such as target reduction and reward amount. We assume that the DSO generates such parameters and certifies the smart meters' signing keys correctly.

Smart Meters (SMs). SMs are embedded IoT devices, installed on users' premises, that measure the electricity flow and periodically transmit the recorded data to the local Energy Aggregator. Some SMs may be corrupted and collude with the Energy Aggregator to reveal private information about their users. We assume that each eligible smart meter holds a valid private signing key with corresponding certification key published along with unique user identity.

Users. Users are the customers of the power system. After evaluating the DR parameters, each user determines whether to take part in the DR program.

Energy Aggregator (EA). The EA collects energy measurements and baselines sent from the SMs. It cooperates with the DR Aggregator to verify that (i) selected users actually provide reduction with respect to the self-reported baseline, (ii) non-selected users were truthful about their baseline report, and (iii) the overall target reduction goal set by the DSO is met. We assume that the EA is trusted with respect to participation privacy (i.e. the anonymization of SM identities), as it does not disclose the link between anonymous public keys and real public keys. However, we assume it is not trusted with respect to data privacy and data integrity.

Demand Response Aggregator (DRA). The DRA is an entity that acts as an intermediary between users and the DSO to enable DR [26]. In our scheme, the DRA is responsible for (i) randomly selecting a subset of users who agreed to participate in the DR program, (ii) computing users' reductions with respect to the self-reported baselines in cooperation with the EA, and (iii) verifying that the target reduction goal is achieved. We assume that the DRA is not trusted for participation privacy, data privacy and data integrity. However, we assume that the DRA does not collude with the EA. If such collusion occurs, participation and data privacy would be compromised but not data integrity.

Bulletin Board (BB). The BB is an append-only log that provides a public view of data for verifying correct execution of all operations. We assume that the BB is not trusted for data privacy or data integrity but provides a consistent view to all parties.

Communication. The EA has secure communication channels with all SMs. The DSO, EA and DRA have a direct and secure communication channel with the BB. The DSO cannot interact directly with the aggregators. The EA and DRA maintain a direct and secure communication channel with each other.

3 High-level description

Our proposed privacy-preserving DR scheme (PPDR) comprises three main phases: (i) initialization, (ii) recruitment, and (iii) activation. The DSO continuously monitors the grid status and, when it detects a condition of high grid stress, it defines a DR program to request users to reduce consumption.

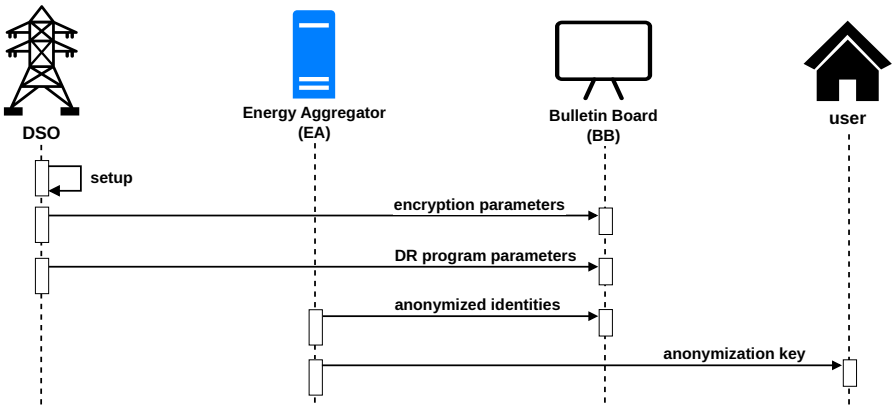


Fig. 1. Sequence diagram of the initialization phase.

Initialization. The DSO generates the encryption scheme parameters and registers all SMs, and the two aggregators. Then, the DSO encrypts and publishes on the BB the DR program parameters, comprising: the DR event start time, DR event duration, reward per unit of reduction, penalty function, selection probability p for taking part in the program, and target overall reduction value for the community managed by the EA. Then, the EA generates anonymized identities for all registered SMs, publishes the resulting list on the BB, and transmits to each user the corresponding anonymization key. Finally, users retrieve DR parameters from the BB and evaluate them independently in order to decide whether to participate or not in the program. Figure 1 shows the sequence diagram of the initialization phase.

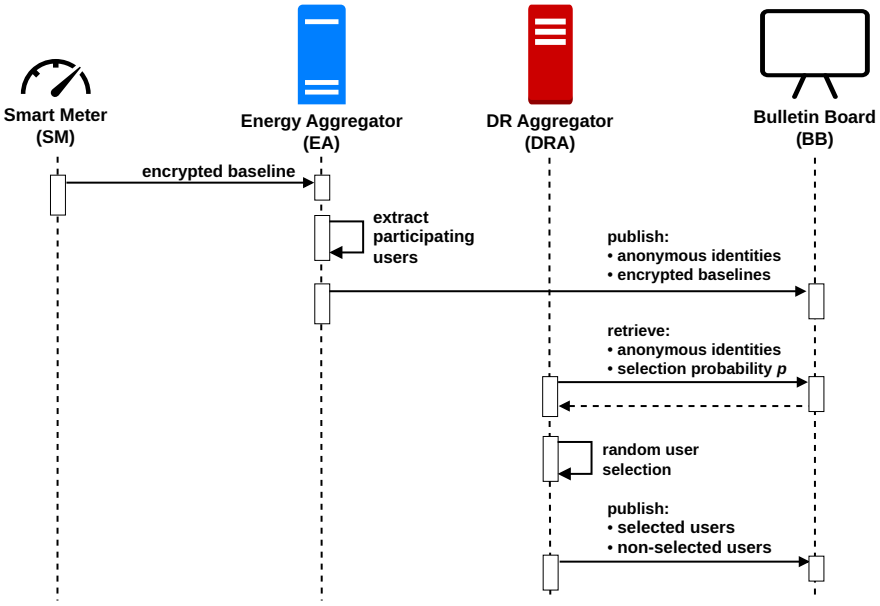


Fig. 2. Sequence diagram of the recruitment phase.

Recruitment. Each SM encrypts the user's baseline energy value before transmitting it to the EA. SMs of users opting to participate in the DR program send a non-null baseline energy value to the EA. This value serves as a reference to determine the amount of reduction when the DR event is activated. Conversely, SMs of users who do *not* intend to participate in the DR program submit a null baseline energy value. Based on the collected values, the EA extracts the set of participating users, and builds the list of their anonymized identities. This list, and the corresponding set of encrypted baselines, are then published on the BB. Then, the DRA retrieves the list of anonymized user identities and the probability of selection p from the BB. The DRA randomly (and verifiably) [17] selects a subset of anonymized participating users with probability p, and publishes the resulting two disjoint lists of selected users and non-selected users on the BB. Finally, each user can verify whether they were selected to provide reduction during the DR event by accessing the BB using the anonymization key obtained during the initialization phase.

At the end of the recruitment phase, users are anonymously partitioned into three disjoint sets: non-participating users, non-selected users and selected users. When the DR event is activated, energy consumption values are evaluated depending on which set the user belongs to. Figure 2 shows the sequence diagram of the recruitment phase.

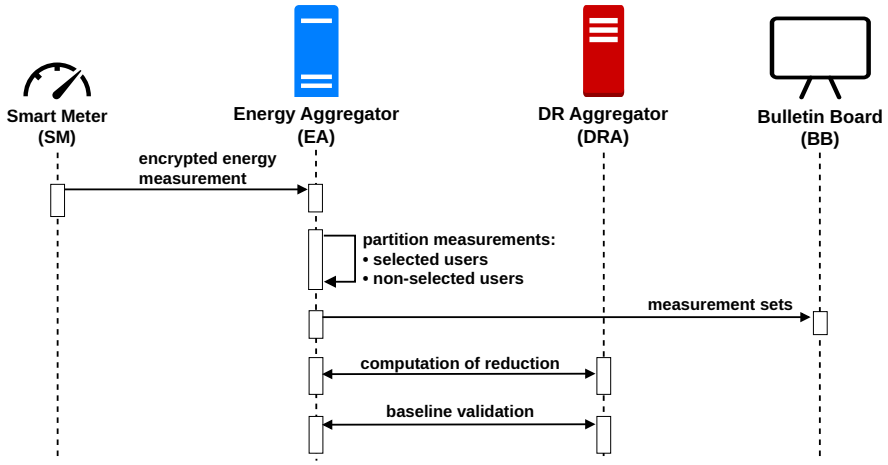


Fig. 3. Sequence diagram of the activation phase.

Activation. The SMs periodically transmit encrypted energy measurements to the EA. Values from non-participating users require no further processing. Based on the user lists on the BB, the EA classifies measurements belonging to selected and non-selected users, generates a measurement set for each category, and publishes the sets on the BB. Then, the computation of load reduction takes place, in which the two aggregators cooperate to verify that, for each selected user, the energy measurement is below the baseline value, signifying a reduction. If this condition is not met, the corresponding user is removed from the selected user set because they are not providing the requested reduction. After evaluating all selected users, the aggregators jointly verify that the aggregated reduction provided by all complying users meets the overall reduction target set by the DSO. If the target is not met, the DRA randomly selects a new subset of users from the non-selected user list, and publishes the updated lists on the BB. Finally, for each non-selected user, the aggregators jointly verify that the energy measurement is consistent with the self-reported baseline, which confirms the truthfulness of the user's report. Figure 3 shows the sequence diagram for the activation phase.

4 Privacy-preserving DR Scheme

Our construction relies on exponential ElGamal encryption [9], a signature scheme, and various NIZKPs. We combine these primitives to realize public key mixing, ciphertext comparison, and encrypted plaintext equality testing. These form the foundation of the main algorithms of the scheme, which are used across the phases introduced in Section 2 and described in detail at the end of this section.

4.1 Cryptographic Primitives

Let λ be a security parameter. Let G be a cyclic group of prime order q generated by generator g. We denote the integers modulo q with $\mathbb{Z}_q$ and write $r \xleftarrow{\$} \mathbb{Z}_q$ for r being chosen uniformly from $\mathbb{Z}_q$.

Exponential ElGamal encryption Our scheme is built on the ElGamal encryption scheme for group G, with generator g of order q and message space $\mathcal{M}$, and the ciphertext space $\mathcal{C} = G \times G$ consisting of the following algorithms.

- $(ek, dk) \leftarrow \mathsf{EKeyGen}(1^\lambda)$: on input the security parameter 1^λ, randomly chooses $dk \xleftarrow{\$} \mathbb{Z}_q$, computes $ek = g^{dk} \in G$. It outputs the encryption key $ek = (p, q, g, ek)$ and the decryption key dk.
- $ct \leftarrow \mathsf{Enc}(ek, m; r)$: on input $m \in \mathcal{M}$, public encryption key ek, and $r \xleftarrow{\$} \mathbb{Z}_q$ computes $C_1 = g^r$ and $C_2 = g^m \cdot ek^r$. It then outputs $ct = (C_1, C_2) \in \mathcal{C}$.
- $m \leftarrow \mathsf{Dec}(dk, ct)$: on input ciphertext $ct = (C_1, C_2)$ and decryption key dk computes $Y = C_2 / C_1^x$, where $Y = g^m$ and outputs $m = \log_g(Y)$ over $\mathcal{M}$.

The encryption scheme has the following properties:

- **Addition of plaintexts.** Given two ciphetexts $ct = (C_1, C_2)$ and $ct' = (C_1', C_2')$ such that $ct = \mathsf{Enc}(ek, m; r)$ and $ct' = \mathsf{Enc}(ek, m'; r')$ the addition of the plaintexts m and m' is computed as follows:

$$\mathsf{Add}(ct, ct') = (C_1 \cdot C_1',\, C_2 \cdot C_2') = \mathsf{Enc}(ek,\, m{+}m';\, r{+}r').$$

- **Multiplication by a public scalar.** A scalar multiplication of given given scalar $\alpha \in \mathbb{Z}_q$ and $ct = (C_1, C_2)$ is computed as follows:

$$\mathsf{SMul}(ct, \alpha) = (C_1^\alpha,\, C_2^\alpha) = \mathsf{Enc}(ek,\, \alpha m;\, \alpha r).$$

- **Re-randomization.** Given $ct = (C_1, C_2)$ a re-randomization with fresh $r' \xleftarrow{\$} \mathbb{Z}_q$ is computed as follows:

$$\mathsf{ReRand}(ek, (C_1, C_2); r') = (C_1 \cdot\ g^{r'},\, C_2 \cdot ek^{r'}) = \mathsf{Enc}(ek,\, m;\, r{+}r').$$

- **Security.** The exponential ElGamal encryption scheme is considered secure against chosen plaintext attacks (IND-CPA), assuming the decisional Diffie-Hellman (DDH) assumption holds in the group G.

Remark 1 *In our scheme we require to subtract the plaintexts corresponding to two given ciphertexts (ct_1, ct_2). Using the homomorphic property, one can define the division of two ciphertexts, which results in an encryption of the difference of the plaintexts modulo q. Note that we assume that plaintext $m_1 \geq m_2$.*

- $ct_{\mathsf{diff}} \leftarrow \mathsf{Diff}(ct_1, ct_2)$: *on input of two ciphertexts $(ct_1 ct_2)$ where $ct_1 = (g^{r_1}, g^{m_1} \cdot ek^{r_1})$ and $ct_2 = (g^{r_2}, g^{m_2} \cdot ek^{r_2})$ compute $ct_{\mathsf{diff}} = ct_1 \cdot ct_2^{-1} = (g^{r_1 - r_2}, g^{m_1 - m_2} \cdot ek^{r_1 - r_2})$ and output ct_{diff}.*

Digital signatures. A signature scheme $\mathsf{Sig} = (\mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is defined as:

- $(sk, pk) \leftarrow \mathsf{SKeyGen}(1^\lambda)$: on input the security parameter 1^λ, sample a secret signing key sk, derive the public key pk, and output (sk, pk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: on input the secret key sk and a message m compute a signature σ on m and output (m, σ).
- $b \leftarrow \mathsf{Verify}(pk, m, \sigma)$: on input the public key pk, message m, and signature σ, output $b \in \{0, 1\}$, where $b = 1$ iff σ is a valid signature on m under pk, and $b = 0$ otherwise.

The signature scheme is assumed EUF–CMA secure.

Public key mixing. Let $\mathbf{pk} = (pk_1, \ldots, pk_N)$ be a set of public keys, where each $pk_i = g^{sk_i}$. A cryptographic mixing (shuffle) [29, 13] of the public keys is another set of group elements $\mathbf{pk'} = (pk'_1, \ldots, pk'_N)$, which contains the same public keys in a re-randomized and permuted order. It is defined as follows.

- $\mathbf{pk'} \leftarrow \mathsf{Mix}(\mathbf{pk}, \mathbf{r'}, \psi)$: on input the set of public keys $\mathbf{pk}$ a permutation $\psi : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$ and random values $\mathbf{r'} = (r'_1, \ldots, r'_N) \in \mathbb{Z}_q^N$, compute $pk'_i = pk_{\psi(i)}{}^{r'_i}$ for all $i = 1, \ldots, N$, and output the set of keys $\mathbf{pk'}$.

Two ciphertexts comparison. Let ct be an encryption of $m \in \{0, 1\}$ and ct' be an encryption of $m' \in \mathbb{Z}_q$. The threshold protocol $CGate(ct, ct')$ [7] allows the owners of the decryption key dk shares to jointly compute an encryption ct'' such that $m \cdot m' \leftarrow Dec(dk, ct'')$. This is done without revealing any plaintexts, and in a verifiable way. In our scheme we use the $CGate$ protocol to compare two encryptions. This comparison protocol [8] is defined as follows.

Let $\mathbf{ct_m} = (ct_1, ct_2, \ldots, ct_b)$ and $\mathbf{ct_{m'}} = (ct'_1, ct'_2, \ldots, ct'_b)$ be the bit-wise ElGamal encryption of m and m'. The algorithm ord compares two bit-wise encryptions as follows:

- $(ct_o, \pi_{ord}) \leftarrow \mathsf{ord}(\mathbf{ct_m}, \mathbf{ct'_m})$: on input two bit-wise encryptions $\mathbf{ct_m}$ and $\mathbf{ct_{m'}}$ compute the comparison according to the algorithm 9 in [8] and output (ct_o, π_{ord}), where ct_o is an encryption of 1 if $m < m'$ and 0 otherwise. The proof of well-formedness and correct decryption π_{ord} ensures the correctness of the comparison.

Proof of signature knowledge. The NIZKP of signature knowledge proves the knowledge of a signature σ on a message m with respect to the public key pk, which is mixed into pk' where $pk' = pk^r$. We define the relation $R_{\sigma,r}$ and the functions as follows:

$$((m, pk'), (\sigma, pk, r)) \in R_{\sigma,r} \text{ iff } \mathsf{Verify}(pk, m, \sigma) = 1 \ \ \wedge \ \ pk' = pk^r$$

- $\pi_{\sigma,r} \leftarrow \mathsf{Proof}_{\sigma,r}(x_{\sigma,r}, \omega_{\sigma,r})$: on input public statement $x_{\sigma,r} = (m, pk')$, secret witness $\omega_{\sigma,r} = (\sigma, pk, r)$, and implicit input $R_{\sigma,r}$ output $\pi_{\sigma,r}$.
- $b \leftarrow \mathsf{Verify}_{\sigma,r}(x_{\sigma,r}, \pi_{\sigma,r})$: on input the public statement $x_{\sigma,r}$, proof $\pi_{\sigma,r}$, and implicit input relation $R_{\sigma,r}$, output $b \in \{0, 1\}$, where $b = 1$ iff $\pi_{\sigma,r}$ is a valid proof, and $b = 0$ otherwise.

Proof of private key knowledge. To prove the NIZKP of private key sk knowledge [25] such that $(sk, pk) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, id)$, the following relation and functions are defined:

$$(pk, sk) \in R_{sk} \text{ iff } pk = g^{sk}$$

- $\pi_{sk} \leftarrow \mathsf{Proof}_{sk}(x_{sk}, \omega_{sk})$: on input the public statement $x_{sk} = (id, pk)$, secret witness $\omega_{sk} = sk$, and implicit input R_{sk}, output π_{sk}.
- $b \leftarrow \mathsf{Verify}_{sk}(x_{sk}, \pi_{sk}))$: on input the public statement x_{sk}, proof π_{sk}, and implicit input relation R_{sk}, output $b \in \{0, 1\}$, where $b = 1$ iff π_{sk} is a valid proof, and $b = 0$ otherwise.

Proof of plaintext knowledge. The NIZKP of plaintext knowledge proves that ct is the correct encryption of a message $m \in \mathcal{M}$ using the public key ek and randomness r. The following relation and functions are defined:

$$((pk, \mathcal{M}, ct), (r, m)) \in R_m \text{ iff } \mathsf{Enc}(pk, m; r) \ \wedge \ m \in \mathcal{M}$$

- $\pi_m \leftarrow \mathsf{Proof}_m(x_m, \omega_m)$: on input the public statement $x_m = (ct, ek)$, secret witness $\omega_m = (m, r)$, and implicit input R_m, output π_m.
- $b \leftarrow \mathsf{Verify}_m(x_m, \pi_m))$: on input the public statement $x_m = (ct, ek)$, proof π_m, and implicit input relation R_m, output $b \in \{0, 1\}$, where $b = 1$ iff π_m is a valid proof, and $b = 0$ otherwise.

Proof of correct public-key mixing. The correctness of a public-key shuffle can be proven by knowledge of the permutation ψ and the set of randomizations $\mathbf{r}' = (r'_1, \ldots, r'_N)$ such that

$$(pk'_1, \ldots, pk'_N) = \mathsf{Mix}((pk_1, \ldots, pk_N), \mathbf{r}', \psi).$$

We define the following relation R_{mix} as follows

$$((\mathbf{pk}, \mathbf{pk}'), (\psi, \mathbf{r}')) \in R_{mix} \text{ iff } \forall i \in \{1, \ldots, N\}, \ pk'_i = pk_{\psi(i)}^{r'_i}$$

- $\pi_{mix} \leftarrow \mathsf{Proof}_{mix}(x_{mix}, \omega_{mix})$: on input the public statement $x_{mix} = (\mathbf{pk}, \mathbf{pk}')$, secret witness $\omega_{mix} = (\psi, \mathbf{r}')$, and implicit input relation R_{mix}, output π_{mix}.
- $b \leftarrow \mathsf{Verify}_{mix}(x_{mix}, \pi_{mix})$: on input the public statement x_{mix}, proof π_{mix}, and implicit input decryption relation R_{mix}, output $b \in \{0, 1\}$, where $b = 1$ iff π_{mix} is a valid proof, and $b = 0$ otherwise.

Proof of correct decryption. To prove that a ciphertext ct decrypts to the plaintext m under the decryption key dk [5], we define the decryption relation R_d and the functions Proof_{dk} and Verify_{dk} as follows:

$$((ek, m, ct), dk) \in R_{dk} \text{ iff } m = \mathsf{Dec}(dk, ct) \wedge pk = g^{dk}$$

- $\pi_{dk} \leftarrow \mathsf{Proof}_{dk}(x_{dk}, \omega_{dk})$: on input the public statement $x_{dk} = (ek, m, ct)$, secret witness $\omega_d = dk$, and implicit input relation R_{dk}, output π_{dk}.
- $b \leftarrow \mathsf{Verify}_{dk}(x_{dk}, \pi_{dk}))$: on input the public statement x_{dk}, proof π_{dk}, and implicit input decryption relation R_{dk}, output $b \in \{0, 1\}$, where $b = 1$ iff π_{dk} is a valid proof, and $b = 0$ otherwise.

Encrypted plaintext equality test (EPET) Given two ciphertexts ct_1 and ct_2 with a homomorphic encryption property, EPET enables computing an encryption of 0 if ct_1 and ct_2 contain the same plaintext, and an encryption of a random group element $g_i \in G$ if the plaintexts are different. We define the following functions:

- $(ct_{eq}, \pi_r) \leftarrow \mathsf{Epet}(ct_1, ct_2)$: on input of two ciphertexts ct_1 and ct_2, chooses r at random ($r \xleftarrow{\$} \mathbb{Z}_q$) and computes $ct_{eq} = \left(\frac{ct_1}{ct_2}\right)^r$ and a NIZKP of knowledge π_r [5] that proves the knowledge r in the relation R_r as follows:

$$((ct_1, ct_2, ct_{eq}), r) \in R_r \text{ iff } ct_{eq} = \left(\frac{ct_1}{ct_2}\right)^r$$

- $\pi_r \leftarrow \mathsf{Proof}_r(x_r, \omega_r)$: on input the public statement $x_r = (ct_1, ct_2, ct_{eq})$, secret witness $\omega_r = r$, and implicit input relation R_r, output π_r.
- $b \leftarrow \mathsf{Verify}_r(x_r, \pi_r)$): on input the statement x_r, proof π_r, and implicit input R_r, outputs $b \in \{0, 1\}$, where $b = 1$ iff π_r is a valid proof, and $b = 0$ otherwise.

4.2 Formal description

The algorithms that define our PPDR scheme are as follows.

$\mathsf{Setup}(1^\lambda, \mathbb{ID}, \mathbb{DR}) \rightarrow (pp)$*:* on input the security parameter 1^λ, the SM identities set $\mathbb{ID}$, and DR program parameters, $\mathbb{DR}$, generate the public parameters pp.

$\mathsf{EKeyGen}(pp) \rightarrow (ek, dk)$*:* on input of the public parameter pp, executes $(ek, dk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ which outputs the encryption key pair. It then computes $\pi_{dk} \leftarrow \mathsf{Proof}_{dk}((pp, ek), dk)$, updates ek such that ek contains pp along with proof of knowledge π_{dk}, and returns (ek, dk).

$\mathsf{SKeyGen}(id, pp) \rightarrow ((id, pk), sk)$*:* on input identity $id \in \mathbb{ID}$ and pp, executes $(sk, pk) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, id)$ to generate the signature key pair (sk, pk). It computes $\pi_{sk} \leftarrow \mathsf{Proof}_{sk}((id, pp, pk), sk)$ and updates pk such that pk contains pp and a proof of knowledge π_{sk}, and returns $((id, pk), sk)$.

$\mathsf{Report}(id, sk, ek, m, t) \rightarrow (pk, (t, \mathbf{ct}, \boldsymbol{\pi_m}, \sigma))$*:* on input identity $id \in \mathbb{ID}$, secret signing key sk, servers public encryption key ek, energy data $m \in \mathcal{M}$, and timestamp t does the following:

1. Compute $m_{\mathsf{bin}} = (m_1, m_2, \ldots, m_b)$ which is the binary representation of m.
2. Compute $ct_i \xleftarrow{\$} \mathsf{Enc}(ek, m_i; r_i)$ for each bit $m_i \in (m_1, m_2, \ldots, m_b)$, and set $\mathbf{ct} = (ct_1, ct_2, \ldots, ct_b)$.
3. Compute $\pi_{m_i} \leftarrow \mathsf{Proof}_m((ct_i, (r_i, m_i))$ for each $ct_i \in \mathbf{ct}$, and set $\boldsymbol{\pi_m} = (\pi_{m_1}, \pi_{m_2}, \ldots, \pi_{m_b})$.
4. Compute $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (t, \mathbf{ct}, \boldsymbol{\pi_m}))$ and return the tuple $(pk, (t, \mathbf{ct}, \boldsymbol{\pi_m}, \sigma))$.

$\mathsf{Anon}_{ID}(\mathbf{pk}) \rightarrow (\mathbf{pk}', \mathbf{r}', \pi_{mix})$*:* on input the set of public keys $\mathbf{pk}$, do as:

1. Compute $\mathbf{pk}' \leftarrow \mathsf{Mix}(\mathbf{pk}, \mathbf{r}', \psi)$.
2. Compute $\pi_{mix} \xleftarrow{\$} \mathsf{Proof}_{mix}((\mathbf{pk}, \mathbf{pk}'), (\mathbf{r}', \psi))$ and return $(\mathbf{pk}', \mathbf{r}', \pi_{mix})$.

$\mathsf{Anonym}(\{(pk_i, t, \mathbf{ct}_i, \boldsymbol{\pi}_{\boldsymbol{m_i}}, \sigma_i)\}_{i\in M}, \{r'_i\}_{i\in M}, sk_{EA}) \rightarrow (\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i\}_{i\in M}, \sigma_{EA})$*:* on input a set of reports $\{(pk_i, t, \mathbf{ct_i}, \sigma_i)\}_{i\in M}$, where $M \subseteq \mathbb{ID}$, a set of secret mixing key $\{r'_i\}_{i\in M}$, the EA secret signing key sk_{EA}, and implicit input the bulletin board $\mathcal{BB}$ does the following:

1. Compute the proof $\pi_i \xleftarrow{\$} \mathsf{Proof}_{\sigma,r}((pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t), (pk_i, \sigma_i, r'_i))$ for each tuple such that $i \in M$. The proof π_i ensures that

$$((\mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t), pk'_i), (\sigma_i, pk_i, r'_i)) \in R_{\sigma,r} \text{ iff}$$
$$\mathsf{Sig.Verify}(pk_i, (\mathbf{ct}_i, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t), \sigma_i) = 1 \quad \wedge \quad pk'_i = pk_i^{r'_i}$$

2. Compute the signature $\sigma_{EA} \xleftarrow{\$} \mathsf{Sign}(sk_{EA}, h_{EA})$, where the hash $h_{EA} = H((pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i)_{i\in M})$, and return the tuple $(\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i\}_{i\in M}, \sigma_{EA})$.

$\mathsf{Validate}(\mathcal{BB}, \{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i\})_{i\in M} \rightarrow \top/\bot$*:* on input the $\mathcal{BB}$, a report $\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i\}$, and implicit input $(ek, \mathbb{ID}, \mathbb{DR}, M)$, checks that

1. $\top \leftarrow \mathsf{Sig.Verify}\big(pk_{EA}, (\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i\})_{i\in M}), \sigma_{EA}\big)$ for each pk'_i such that $i \in M$.
2. no terms in $\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, \pi_i\}$ already appear in $\mathcal{BB}$
3. $\top \leftarrow \mathsf{Verify}_{\sigma,r}((pk'_i, \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i)$ and $\top \leftarrow \mathsf{Verify}_m(\mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}})$.

If any of the checks fail, it returns $\bot$ otherwise $\top$.

$\mathsf{Eval}(\mathcal{BB}) \rightarrow \mathcal{BB}$*:* On input of bulletin board $\mathcal{BB}$ retrieves the baseline ciphertext $\mathbf{ct}_b$, the energy consumption ciphertext $\mathbf{ct}_m$ for each $pk' \in \mathbf{pk}'$ at timestamp t and uses decryption key shares, which together implicitly define dk to compute:

1. $(ct_o, \pi_{ord}) \leftarrow \mathsf{ord}(\mathbf{ct_m}, \mathbf{ct_b})$, which on input two ciphertexts in binary form $(\mathbf{ct_b}, \mathbf{ct_m})$, computes ct_o and the proof of correct computation π_{ord}. It then returns (ct_o, π_{ord}), where ct_o is an encryption of 1 if $m < b$, and 0 otherwise.
2. $(\mathbf{ct_{red}}, \pi_{dec}, 1) \leftarrow \mathsf{Reduct}(\mathbf{ct_b}, \mathbf{ct_m}, ct_o)$, which on input the ciphertexts $\mathbf{ct_b}$, $\mathbf{ct_m}, ct_o$, and implicit input of dk, computes $m \leftarrow \mathsf{Dec}(dk, ct_o)$ and $\pi_{dec} \leftarrow \mathsf{Proof}_{dec}((ct_o, m), dk)$. If $m = 1$, compute $\mathbf{ct_{red}} \leftarrow \mathsf{Diff}(\mathbf{ct_b}, \mathbf{ct_m})$ and then output $(\mathbf{ct_{red}}, \pi_{dec}, 1)$. Otherwise, output $(\bot, \pi_{dec}, 0)$.
3. $ct_{\mathsf{sum}} \leftarrow \mathsf{Agg}(\mathbf{CT_{red}})$, which on input the set $\mathbf{CT_{red}} = \{(\mathbf{ct_{red}}, t, pk')\}_{pk'\in\mathbf{pk}'}$ computes the encrypted aggregation $\mathbf{ct} = \prod_{\mathbf{ct_i}\in\mathbf{CT_{red}}} \mathbf{ct_i}$[5], which is in binary form. It then converts $\mathbf{ct}$ into a single ciphertext ct_{sum}, which contains an encryption of the corresponding aggregated plaintexts, and returns ct_{sum}.

[5] Recall that $\mathbf{ct_i} = (ct_{i_1}, ct_{i_2}, \ldots, ct_{i_b})$ and therefore the encrypted aggregation $\mathbf{ct} = (\prod_{i=1}^{k} ct_{i_1}, \prod_{i=1}^{k} ct_{i_2}, \ldots, \prod_{i=1}^{k} ct_{i_b})$, where $k = |\mathbf{CT_{red}}|$.

4. $(\mathbf{ct_{eq}}, \boldsymbol{\pi}_{eq}) \leftarrow \mathsf{Pet}(ct_{\mathsf{sum}}, \mathbf{ct}_T)$ which on input ct_{sum} and the set of target ciphertexts $\mathbf{ct}_T$, computes $(ct_{eq_i}, \pi_{r_i}) \leftarrow \mathsf{Epet}(ct_{\mathsf{sum}}, ct_{T_i})$ for each $ct_{T_i} \in \mathbf{ct_T}$. The ciphertext ct_{eq_i} is an encryption of 0 if the plaintexts of ct_{sum} and ct_{t_i} are equal, and an encryption of a random group element $g_i \in G$ otherwise. It sets $\boldsymbol{\pi_{eq}} = \{\pi_{r_i}\}_{i \in T}$ and $\mathbf{ct_{eq}} = \{ct_{eq_i}\}_{i \in T}$ It returns $(\mathbf{ct_{eq}}, \boldsymbol{\pi}_{eq})$.
5. $(\boldsymbol{R}, \boldsymbol{\pi}_{dec}) \leftarrow \mathsf{PDec}(\mathbf{ct}_{eq}, dk)$ which, for each $ct_{eq_i} \in \mathbf{ct}_{eq}$ and decryption key shares of dk, computes $r_i \leftarrow \mathsf{Dec}(dk, ct_{eq_i})$. Note that r_i is either g^0 or a random element $g_i \in G$. It also computes the proof $\pi_i \xleftarrow{\$} \mathsf{Proof}_{dec}(ek, r_i, ct_{eq_i}), dk)$. It sets $\boldsymbol{R} = (r_i)_{i \in T}$, $\boldsymbol{\pi_{dec}} = (\pi_i)_{i \in T}$, and returns $(\boldsymbol{R}, \boldsymbol{\pi_{dec}})$.

$\mathsf{Verify}(\mathcal{BB}, \boldsymbol{R}, \Pi) \rightarrow \top/\bot$*:* On input $\mathcal{BB}$, the evaluation result $\boldsymbol{R}$, and the corresponding proofs $\Pi = (\pi_{ord}, \boldsymbol{\pi_{dec}}, \boldsymbol{\pi_{eq}}, (\pi_i)_{i \in T})$ verifies the following proofs: The proof of evaluation result Π, all ZKPs π_{sk}, π_{dk}, the proof of public key mixing π_{mix}, the wellformness proof $\boldsymbol{\pi_{m_i}}$ and the proof of the signature knowledge π_i of each report $\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi_{m_i}}, t, \pi_i\}$, and the signature σ_{EA} on $\mathcal{BB}$. If any of the proof checks fail, it returns $\bot$ otherwise $\top$.

4.3 Description of the Phases

Initialization. The DSO executes $\mathsf{Setup}(1^\lambda, \mathbb{ID}, \mathbb{DR})$ to initialize the scheme by generating all public parameters for setting up signing, encryption, and ZKPs. The EA and DRA execute $\mathsf{EKeyGen}(pp)$ to generate their encryption key pairs (ek_{EA}, dk_{EA}) and (ek_{DRA}, dk_{DRA}), respectively. The DSO publish their combined encryption key $ek = ek_{EA} \cdot ek_{DRA}$ on the bulletin board $\mathcal{BB}$.

The DSO, the SMs, and the two aggregators independently execute the algorithm $\mathsf{SKeyGen}(id, pp)$ to generate their signing key pairs. The DSO collects all the corresponding public keys, verifies the identity id of each SM, as well as the NIZKP contained within the SM public key pk. Note that such proofs are needed to prevent replay attacks [23]. If the verification is successful, the DSO includes SM into the list of registered smart meters $\mathbf{ID_{pk}} = \{(id, pk)\}_{id \in \mathbb{ID}}$, signs $\mathbf{ID_{pk}}$, and publishes it on $\mathcal{BB}$. Furthermore, the DSO publishes the signed list of the two aggregators' public keys on $\mathcal{BB}$.

The DSO generates a noisy set of integers $\{1, \cdots, T-1, 0, \cdots\}$ that consists of the $T-1$ integers less than the reduction target value T, extended with a number of zeros that serve as noise to prevent potential leakage of the target value based on the list's size. The DSO then shuffles and encrypts the list, referred to as the reduction target list[6] $\mathbf{ct_T}$, and publishes it on $\mathcal{BB}$. The EA executes $\mathsf{Anon}_{\mathsf{ID}}(\mathbf{pk})$ to generate a mixed and anonymized public key set $\mathbf{pk}'$ from the given $\mathbf{ID_{pk}}$. It computes $\mathbf{r}'$ and π_{mix}, where the proof π_{mix} verifies the correctness of the

[6] The target list $\mathbf{ct_T}$ is intended for use by the aggregators during evaluation. Because additively homomorphic encryption does not support comparison of encrypted integers, the aggregators cannot directly test whether the encrypted aggregate is below the reduction target. We therefore require the DSO to precompute a noisy list of ciphertexts corresponding to all values less than the target, which enables evaluation of the encrypted reduction without revealing the underlying values.

anonymization process, and $\mathbf{r}'$ represents the secret randomization used during the anonymization. The EA publishes the anonymized public keys $(\mathbf{pk}', \pi_{mix})$ on $\mathcal{BB}$ and sends $g^{r'}$ to the corresponding SM with identity (id, pk).

Recruitment. All SMs generate an encrypted energy baseline value $(pk, (t, \mathsf{ct}, \sigma))$ using the algorithm $\mathsf{Report}(id, sk, ek, m, t)$ and send them to EA. Users wishing to participate in the DR program should send their non-zero baseline value, while non-participating users should send a deterministic encryption of zero. By sending the encrypted baselines through secure channels, an adversary cannot identify whether a smart meter participates in or not in the DR program. The EA extracts the participating SMs P that sent a non-zero baseline value. Then, for all extracted SMs, the EA executes the anonymization algorithm $\mathsf{Anonym}(\{(pk_i, t, \mathbf{ct}_{b_i}, \pi_{\mathbf{b_i}}, \sigma_{\mathbf{i}})\}_{\mathbf{i}\in\mathbf{P}}, \{\mathbf{r}'_{\mathbf{i}}\}_{\mathbf{i}\in\mathbf{P}}, \mathbf{sk_{EA}})$ to compute $(\{pk'_i, \mathbf{ct_{b_i}}, \boldsymbol{\pi}_{\boldsymbol{b_i}}, t, \pi_i\}_{i\in P}, \sigma_{EA})$ containing the encrypted baseline for each participant. Recall that, for each baseline report $(pk_i, (t, \mathbf{ct_{b_i}}, \boldsymbol{\pi}_{\boldsymbol{b_i}}, \sigma_i))$, Anonym selects only $(\mathbf{ct_{b_i}}, \boldsymbol{\pi}_{\boldsymbol{b_i}}, t)$ and provides a proof that ensures that the aggregator knows a valid signature on $(\mathbf{ct_{b_i}}, \boldsymbol{\pi}_{\boldsymbol{b_i}}, t)$ signed by pk_i. It then publishes all $(pk'_i, \mathbf{ct_{b_i}}, t, \boldsymbol{\pi}_{\boldsymbol{b_i}}, \pi_i)$ on $\mathcal{BB}$. The EA publishes on $\mathcal{BB}$ the anonymized public keys $M = \{pk'_1, pk'_2, \ldots, pk'_m\}$. Each participant SM can find their pk'_i on $\mathcal{BB}$.

The DRA selects a small, random subset $S \subseteq M$ based on a selection probability $p \in \mathbb{DR}$. The DRA then signs and publishes S on $\mathcal{BB}$.

Activation. The EA periodically collects all energy measurement reports $\{\mathbf{ct}_{m_i}\}$ from all smart meters. It creates two sets of reports: one for the selected SMs for reduction S, and the other one for the non-selected users $NS = M \setminus S$. Then, it anomymizes the two sets by executing the algorithm Anonym on each of them. The EA then publishes on $\mathcal{BB}$ the resulting two anonymized sets.

For each SMs consumption reports in S and NS, both aggregators jointly run the algorithm $\mathsf{Eval}(\mathcal{BB})$ on $\mathcal{BB}$ using their shares of dk to evaluate the following:

1. For each SMs in M, the aggregators jointly evaluate whether the consumption report value is less than the baseline report value or not, without knowing the individual report value, but using the algorithm $\mathsf{ord}(\mathbf{ct_{m_i}}, \mathbf{ct_{b_i}})$. According to the output (ct_o, π_{ord}) of the algorithm ord, the aggregators mark the anonymized public keys whose consumption is not less than the baseline report, which are the SMs that failed to meet the DR program requirement.
2. For the set of public keys in S that were not marked in the previous step, the encrypted individual reduction values $\mathbf{CT_{red}}$ are computed by the DRA using the algorithm $\mathsf{Reduct}(\mathbf{ct_b}, \mathbf{ct_m}, ct_o)$. The encrypted individual reductions are then aggregated by applying the algorithm $ct_{sum} \leftarrow \mathsf{Agg}(\mathbf{CT_{red}})$. To verify that the encrypted aggregate reduction aligns with the hidden target value, the encrypted aggregated result ct_{sum} and the list of target ciphertexts $\mathbf{ct}_T$ are compared using the algorithms $\mathsf{Pet}(ct_{\mathsf{sum}}, \mathbf{ct}_T)$ and $\mathsf{PDec}(\mathbf{ct}_{eq}, dk)$. Note that none of these algorithms reveals any actual values, but only (i) whether the consumption is less than the baseline report and (ii) whether

the hidden target value has been reached. All the computations can be publicly verified using the algorithm Verify. All the computations can be publicly verifies using the algorithm Verify.

5 Security Analysis

In the following, we state our claims and provide intuitions on the security and privacy of our proposed scheme. We discuss how our scheme achieves smart meter data privacy, participation privacy, and public verifiability[7]. We refer the reader to Appendix A for the details about definitions and proofs.

Theorem 1 (smart meter data privacy). *Assuming the underlying encryption scheme is IND-CCA1 secure, the signature scheme is EUF-CMA secure, and the ZKP system satisfies ZKP of knowledge properties, our PPDR scheme achieves smart meter data privacy as defined in Definition 1.*

Smart meter data privacy of a DR scheme ensures that an adversary who has access to the consumption of corrupted SMs cannot learn the consumptions of honest SMs beyond what can be inferred from the evaluation during the activation phase. The EA is an (internal) adversary who cannot decrypt the individual reports of honest SMs. However, the EA knows which anonymized ciphertext corresponds to which honest SM identity. The IND-CCA1 property of the encryption scheme ensures that the adversary cannot submit a new report constructed from the ciphertext submitted by an honest SM, and the NIZKP used in the encryption scheme prevents the manipulation of the ciphertext submitted by the honest SM. Moreover, due to the EUF-CMA of the signature scheme, the adversary cannot submit a ciphertext on behalf of the honest SM. During the evaluation in the activation phase, the encrypted reduction value is compared by the encrypted target values using PET. Note that the target values are encrypted by DSO, and the comparison only reveals whether the aggregated reduction value meets the target value or not. Knowing the reduction value of an honest SM only reveals the difference between baseline and consumption energy data values. Therefore, even if the the target values are not encrypted, an adversary may learn the reduction of honest SMs but would still require additional information to recover their individual baseline or consumption values.

Theorem 2 (participation privacy.). *Our PPDR scheme achieves participation privacy according to Definition 3 if the signature is EUF-CMA and assuming the ZKP of knowledge properties.*

Participation privacy ensures that an adversary who knows the total number of the participating SMs and the result of the activation phase cannot learn whether an honest SM participated or not at the DR program. We assume that

[7] Public verifiability is not primarily for the benefit of individual clients, but rather to guarantee system-wide integrity, auditability, and robustness against misbehavior by any party, including the aggregators.

the EA is trusted and that the adversary (i.e. DRA) can corrupt a subset of SMs and decrypt their anonymized report ciphertexts in order to try to identify an honest SM that participates in the DR program. Given the SM public key $pk \in \mathbf{pk}$, the EA generates an anonymized set $(\mathbf{pk}', \pi_{mix})$ and sends to each SM the corresponding anonymization key g^r. The SM can recover their anonymization key $pk' \in \mathbf{pk}'$, using their key sk, where $pk' = (g^r)^{sk} = pk^r$. Since the communication channel between the EA and the SM is secure, without knowing g^r, the pair (pk, pk') is identically distributed to (pk, pk'') for any $pk'' \in \mathbf{pk}'$. Distinguishing between these two distributions is information-theoretically hard.

Even if the adversary knows g^r, we can still prove participation privacy under the DDH assumption: the adversary cannot distinguish (g^r, g^{sk}, pk'') from (g^r, g^{sk}, pk'), and $pk' = g^{sk \cdot r}$. Since the signature scheme is EUF-CMA, the adversary cannot submit a report on behalf of honest SMs. The zero-knowledge property of the proof of signature knowledge π_i in the anonymized report guarantees that the proof reveals no information about the witness, including the SM public key or the underlying signature.

Theorem 3 (verifiability.). *Our PPDR scheme achieves public verifiability according to Definition 4 if the signature is EUF-CMA and assuming the soundness of the ZKP, and the correctness of the encryption scheme.*

Our scheme provides public verifiability, meaning that anyone can check the correctness of the anonymization process and the result computed at the end of the activation phase. This verification process ensures that the published result accurately corresponds to what would be obtained by decrypting each report from the BB and then applying the evaluation function in the activation phase to the corresponding plaintexts. Given a subset of honest SMs that verify the $\mathcal{BB}$ after their participation, an adversary who controls a subset of corrupted SMs, and both aggregators has a negligible advantage in creating a BB and evaluation results that do not meet the following properties: 1) all selected honest SMs who have verified their reports are included in the evaluation process in the activation. 2) For non-verifying selected honest SMs, the adversary controlling the BB can at most delete such SMs from the set of selected SMs. 3) No more than two reports can be associated to a selected corrupted SM.

Given the $\mathcal{BB}$ and the sets R and Π, one can publicly verify that all submitted reports lie in the valid message space, which follows from the soundness of the well-formedness proofs. The soundness of the NIZKP of signature knowledge guarantees that each report was generated by a registered and selected SM. The soundness of the mixing NIZKP ensures the correctness of the public-key mixing. Finally, combining the soundness of the decryption procedure, PET, and the correctness and homomorphic properties of the encryption scheme, we conclude that the published results correspond exactly to the computations performed on the underlying plaintext inputs. The EUF-CMA of the signature scheme and the soundness of the NIZKP of signature knowledge ensure that the adversary cannot modify the honest SM reports or submit a report on behalf of them.

# selected	10	20	30	40	50	60	70	80	90	100
# total	100	200	300	400	500	600	700	800	900	1000
Init (s)	72	144	215	288	360	427	502	576	645	716
Recruit (s)	28	53	80	107	133	160	189	217	237	262
Activation (s)	**3**	**6**	**10**	**12**	**16**	**18**	**23**	**24**	**27**	**29**

Table 1. Runtime of the three protocol phases as a function of the number of users. '# selected' denotes the number of selected users (10% of the total users).

6 Performance

We implemented a prototype of our protocol in Python[8]. All experiments were conducted on a MacBook Pro (2021) with an Apple M1 Max processor (10 cores: 8 performance and 2 efficiency) and 64 GB RAM. The implementation is structured as a modular library that exposes the main cryptographic primitives (encryption, homomorphic operations, and signature generation/verification) together with the higher-level protocol logic.

For elliptic-curve operations, we rely on the Charm cryptographic framework [2], which provides an efficient C/OpenSSL backend for big-integer and group operations. This allows us to prototype our design quickly while still benefiting from optimized low-level arithmetic. Our choice of Python greatly simplifies prototyping but comes at a performance cost. Our framework uses Schnorr signature scheme significantly and it is implemented without a native backend. Therefore, we expect that a production-grade re-implementation in Rust or C, using the same cryptographic design but with compiled, constant-time curve arithmetic, would yield substantially better performance. In this paper, our goal is to demonstrate that the proposed protocol is practical.

Benchmarks. We benchmark the protocol by varying the number of users that wish to participate in the DR program and measuring the execution time of each protocol phase: initialization, recruitment, and activation. In all experiments, the DR Aggregator selects a subset of users equal to 10% of the users that registered for the event (e.g., 10 selected out of 100 total, 20 out of 200, and so on). We explicitly exclude network and communication overhead to isolate the cost of the cryptographic operations. From a system perspective, initialization and recruitment are not time-critical. Activation is the performance-critical phase. Activation starts at the published DR start time and includes the main cryptographic operations on newly generated measurement data (e.g., consumption and reduction reports), which may need to be processed every few minutes. For this reason, we pay particular attention to how the activation time scales with the number of selected users.

We vary the total number of users from 100 to 1000 in steps of 100 (with 10% selected by the DR Aggregator in each experiment) and measure the wall-clock execution time of each phase. The results are summarized in Table 1. The

[8] https://github.com/emad7105/ppdr/

total runtime grows almost perfectly linearly with the number of selected users: a least-squares fit of the total time (not shown in the table) gives an increase of approximately 1 second per additional selected user (about 10 seconds per additional 10 users), with goodness of fit $R^2 \approx 0.999$. Initialization and recruitment also exhibit linear scaling, with slopes of roughly 0.72 and 0.26 seconds per user, respectively, while activation grows at about 0.03 seconds per user. This confirms that the computational overhead of the cryptographic routines scales linearly in the number of participating users, as expected from the protocol design.

Given that initialization and recruitment can be scheduled far in advance of the DR event, the dominant real-time cost is activation. Our measurements show that even with 1000 users (100 selected), the activation phase completes in well under half a minute, which is compatible with practical DR scenarios where measurements are processed at intervals of several minutes. Koltunov et al. [14] report that the average size of an Energy Community within the European Union is below 800 members (excluding Belgium and Spain outliers). Therefore, our experimental setting is close to a real-world setup.

7 Related Work

Different strategies for tackling privacy in smart grid and DR scenarios have been proposed [24, 16, 15, 11, 10]. In particular, several approaches have been introduced to protect user data in DR programs, including Privacy-Preserving Data Aggregation (PPDA), Differential Privacy (DP), and blockchain-based schemes.

PPDA-based. Palazzo et al. [21] proposed mPVAS, a publicly verifiable PPDA protocol. The model remains secure even in the presence of malicious users who collude with a malicious aggregator, and extensions were developed to identify malicious users and handle dropouts. However, it does not support attributable data, which is an essential aspect in incentive-based DR programs to compute individual contributions and associated rewards. Sui and de Meer [27] proposed 'BAP', a PPDA scheme based on homomorphic encryption that enables users to prove the amount of provided reduction without disclosing private metering data by leveraging range proofs. In this scheme, the reward does not depend on the actual reduction, which negatively impact fairness, and the baseline is assumed to equal the measurement immediately before the DR event, which is not generally valid.

DP-based. Zaman et al. [30] utilized a Conditional GAN to generate DP synthetic smart meter data. By adding noise to real measurements before training the GAN, the method produces privacy-preserving synthetic datasets that mitigate adversarial input attacks and user re-identification attacks. While this approach enables DR-related optimization without relying on trusted aggregators, its ability to provide privacy-preserving DR functionalities is limited because participation privacy is not ensured, and verification of user reductions is not supported. Chen et al. [6] proposed a DP-based strategy for a recommendation system in DR programs that supports privacy-preserving selection of

the optimal customer for providing reduction. This approach requires a trade-off between privacy level and utility, and participation privacy is not guaranteed.

Blockchain-based. Park et al. [22] proposed a blockchain-based scheme for DR programs that provides privacy-preserving authentication and data integrity. While the use of blockchain ensures data integrity and transparency, the scheme suffers from overhead because SMs must commit the same data twice. Furthermore, the DR operator can decrypt metering data, which compromises privacy. Abouyoussef and Ismail [1] introduced a blockchain-based approach that uses group signatures and a decentralized random number generation algorithm to provide user anonymity and data unlinkability. However, the model assumes that all entities are honest and that no collusion occurs.

8 Conclusions

In this paper we propose the first scheme for privacy-preserving demand response that simultaneously achieves: 1) smart meter data privacy; 2) participation privacy; and 3) public verifiability. Our scheme relies on two key insights. The first is the use of bitwise encryption combined with a threshold-comparison protocol, enabling aggregators to compute publicly verifiable consumption-reduction values without accessing individual energy data. The second insight is the generation of anonymized public keys for registered smart meters together with NIZKP of signature knowledge, ensuring participation privacy and allowing the corresponding computations to be veriably outsourced to aggregators.

Our scheme allows voluntary participation in DR programs, while enabling the system operator to set a target reduction value that can be enforced and verified at the end of the DR event. It shows that strong privacy, verifiability guarantees, and practical performance are not mutually exclusive and can be achieved simultaneously in DR programs.

References

1. Abouyoussef, M., Ismail, M.: Blockchain-based networking strategy for privacy-preserving demand side management. In: ICC. IEEE (2021)
2. Akinyele, J.A., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering (2013)
3. Casella, E., Silvestri, S., Baker, D.A., Das, S.K.: A human-centered power conservation framework based on reverse auction theory and machine learning. ACM Transactions on Cyber-Physical Systems (2024)
4. Celik, Z.B., Babun, L., Sikder, A.K., Aksu, H., Tan, G., McDaniel, P., Uluagac, A.S.: Sensitive information tracking in commodity IoT. In: USENIX Security. USENIX Association (2018)
5. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO. Springer (1992)
6. Chen, W., Zhou, A., Zhou, P., Gao, L., Ji, S., Wu, D.: A privacy-preserving online learning approach for incentive-based demand response in smart grid. IEEE Systems Journal (2019)

7. Cortier, V., Gaudry, P., Yang, Q.: A toolbox for verifiable tally-hiding e-voting systems. In: ESORICS. Springer (2022)
8. Cortier, V., Gaudry, P., Yang, Q.: Is the JCJ voting system really coercion-resistant? In: CSF. IEEE (2024)
9. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory (1985)
10. Erkin, Z.: Private data aggregation with groups for smart grids in a dynamic setting using CRT. In: WIFS. IEEE (2015)
11. Erkin, Z., Veugen, T.: Privacy enhanced personal services for smart grids. In: SEGS. ACM (2014)
12. Good, N., Ellis, K.A., Mancarella, P.: Review and classification of barriers and enablers of demand response in the smart grid. Renewable and Sustainable Energy Reviews (2017)
13. Haenni, R., Locher, P., Koenig, R., Dubuis, E.: Pseudo-code algorithms for verifiable re-encryption mix-nets. In: FC. Springer (2017)
14. Koltunov, M., Pezzutto, S., Bisello, A., Lettner, G., Hiesl, A., van Sark, W., Louwen, A., Wilczynski, E.: Mapping of Energy Communities in Europe: status quo and review of existing classifications. Sustainability (2023)
15. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-friendly aggregation for the smart-grid. In: PETS. Springer (2011)
16. Leontiadis, I., Molva, R., Önen, M.: Privacy preserving statistics in the smart grid. In: ICDCSW. IEEE (2014)
17. Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: FOCS (1999)
18. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: BuildSys. ACM (2010)
19. Muthirayan, D., Baeyens, E., Chakraborty, P., Poolla, K., Khargonekar, P.P.: A minimal incentive-based demand response program with self reported baseline mechanism. IEEE Transactions on Smart Grid (2019)
20. Ozmen, M.O., Farrukh, H., Celik, Z.B.: Physical side-channel attacks against intermittent devices. Proceedings on Privacy Enhancing Technologies (2024)
21. Palazzo, M., Dekker, F.W., Brighente, A., Conti, M., Erkin, Z.: Privacy-preserving data aggregation with public verifiability against internal adversaries. In: USENIX Security. USENIX Association (2024)
22. Park, K., Lee, J., Das, A.K., Park, Y.: BPPS: Blockchain-enabled privacy-preserving scheme for demand-response management in smart grid environments. IEEE Transactions on Dependable and Secure Computing (2022)
23. Rakeei, M., Giustolisi, R., Lenzini, G.: Secure internet exams despite coercion. In: DPM: ESORICS Workshops. Springer (2022)
24. Rial, A., Danezis, G., Kohlweiss, M.: Privacy-preserving smart metering revisited. International Journal of Information Security (2018)
25. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology (1991)
26. Siano, P.: Demand response and smart grids—a survey. Renewable and Sustainable Energy Reviews (2014)
27. Sui, Z., de Meer, H.: BAP: A batch and auditable privacy preservation scheme for demand response in smart grids. IEEE Transactions on Industrial Informatics (2019)
28. Wang, Z., Lu, B., Wang, B., Qiu, Y., Shi, H., Zhang, B., Li, J., Li, H., Zhao, W.: Incentive based emergency demand response effectively reduces peak load during heatwave without harm to vulnerable groups. Nature Communications (2023)

29. Wikström, D.: User manual for the Verificatum Mix-Net, Version 1.4.0 (2013)
30. Zaman, Z., Gauravaram, P., Jha, S., Hu, W.: Privacy-preserving smart metering through data synthesis with differential privacy. In: BuildSys. ACM (2025)

A Definitions and Security Proofs

A.1 Smart Meter Data Privacy

We define *smart meter data privacy* for a privacy-preserving DR scheme in terms of a game between a challenger and an adversary $\mathcal{A}$. The goal is to ensure that a secure DR scheme does not reveal information about individual smart meter data beyond what can be inferred from the evaluation results. The DRA is assumed not to collude with the EA. We assume that $\mathcal{A}$ namely the EA colludes with corrupted smart meters and knows the link between each anonymized ciphertext and its corresponding public key. The adversary $\mathcal{A}$ aims to learn the secret energy data that is submitted by an honest smart meter.

We formalize this idea as a game in which $\mathcal{A}$ distinguishes between two scenarios: the real scenario and the fake one. In the real scenario, $\mathcal{A}$ has access to a list containing ciphertexts generated by both honest and corrupted smart meters, along with the corresponding evaluation result. In the fake scenario, $\mathcal{A}$ views a fake list instead of the real one, while the evaluation results are still computed based on the real list. We also explicitly model the additional information alongside the results (e.g., a proof of correct computation Π). We require that this additional information does not reveal any additional information about individual data. Specifically, we require a simulator that can generate additional information corresponding to the real scenario result with respect to the fake list.

We model the *smart meter data privacy* game through $Exp^{DRPRIV,b}_{\mathcal{A},\mathcal{DR}}$ defined in Figure 4. We initialize two empty lists $\mathcal{B}_0$ and $\mathcal{B}_1$. The list $\mathcal{B}_0$ corresponds to the real experiment $Exp^{DRPRIV,0}_{\mathcal{A},\mathcal{DR}}$, where the evaluation result (R, Π_0) is computed on $\mathcal{B}_0$. The adversary $\mathcal{A}$ has access to $\mathcal{B}_1$ in experiment $Exp^{DRPRIV,1}_{\mathcal{A},\mathcal{DR}}$ where the evaluation result R is the same as the one computed on $\mathcal{B}_0$. However, the corresponding proof Π_1 is simulated. The experiment begins with a setup phase that initializes the DR scheme's public parameters and the public keys of the registration phase. The adversary $\mathcal{A}$ then receives $(ek, \mathbf{pk}, \mathbf{r'mix}, \{sk_i\}_{i\in\mathsf{Corr}})$, which contains the encryption key ek, the registered smart meter public keys $\mathbf{pk}$, the secret mixing keys $\mathbf{r'_{mix}}$, and the secret signing keys of the corrupted smart meters $\{sk_i\}_{i\in\mathsf{Corr}}$ as well as access to the

$\mathcal{O}$**View:** This oracle allows the adversary to see the publishable part of $\mathcal{B}_b$ in experiment $Exp^{DRPRIV,b}_{\mathcal{A},\mathcal{DR}}$. The oracle returns publish($\mathcal{B}_b$).

$\mathcal{O}$**ReportIND:** The oracle ReportIND takes two potential values (m_0, m_1) for user id at timestamp t. It computes the corresponding reports Rt_0 and Rt_1 and appends them to $\mathcal{B}_0$ and $\mathcal{B}_1$, respectively.

$\mathcal{O}$**Append:** This oracle allows the adversary to create $Rt_{\mathcal{A}}$ on behalf of any user. If $Rt_{\mathcal{A}}$ is valid with respect to $\mathcal{B}_b$, it is appended to both lists.

$\mathcal{O}$**Eval:** This oracle allows the adversary to view the evaluation result of the privacy-preserving DR scheme (PPDR). In both experiments, the result is obtained by evaluating on $\mathcal{B}_0$. In experiment $Exp^{\mathsf{DRPRIV},0}_{\mathcal{A},\mathcal{DR}}$, the additional information produced by the aggregators is given to $\mathcal{A}$, whereas in the second experiment, this additional information is simulated.

The adversary $\mathcal{A}$ can call oracles $\mathcal{O}$View, $\mathcal{O}$ReportIND, $\mathcal{O}$Append in any order, and any number of times. The adversary $\mathcal{A}$ can call the oracle $\mathcal{O}$Eval once, and returns a guess bit b' after receiving the evaluation result from $\mathcal{O}$Eval. The adversary $\mathcal{A}$ wins the game if $b' = b$.

Definition 1 (DRPRIV). *Consider algorithms* (Setup, Report, Eval, Publish, Verify) *for a set* $\mathbb{ID}$ *of identities and the DR parameters* $\mathbb{DR}$. *We say that the DR scheme achieves smart meter data privacy if there exists a simulator algorithm sim-proof such that the advantage of any probabilistic polynomial-time (PPT) adversary* $\mathcal{A}$ *in* $Exp^{DRPRIV,b}_{\mathcal{A},\mathcal{DR}}$,

$$\left| \Pr\left[Exp^{DRPRIV,0}_{\mathcal{A},\mathcal{DR}}(1^\lambda) = 1 \right] - \Pr\left[Exp^{DRPRIV,1}_{\mathcal{A},\mathcal{DR}}(1^\lambda) = 1 \right] \right|$$

is negligible in λ.

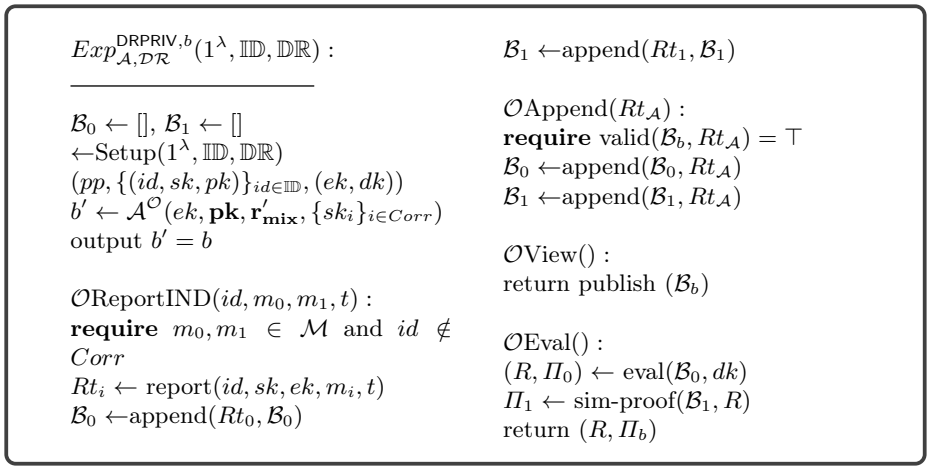


Fig. 4. The game-based security definition of smart meter data privacy $Exp^{DRPRIV,b}_{\mathcal{A},\mathcal{DR}}(1^\lambda, \mathbb{ID}, \mathbb{DR})$. The adversary wins if it guesses the bit b. The oracle $\mathcal{O}$ReportIND either inputs the left report m_0 or the right one m_1 based on the bit b. The label ℓ can be interpreted as identification.

Definition 2 (IND-CCA1). *The advantage of an adversary* $\mathcal{A}$ *in the IND-CCA1 game is defined as*

$$Adv^{\boldsymbol{IND-CCA1}}_{\mathcal{A}} = \left| \Pr[Exp^{IND-CCA1,0}_{\mathcal{A}}(1^\lambda)] - \Pr[Exp^{IND-CCA1,1}_{\mathcal{A}}(1^\lambda)] \right|$$

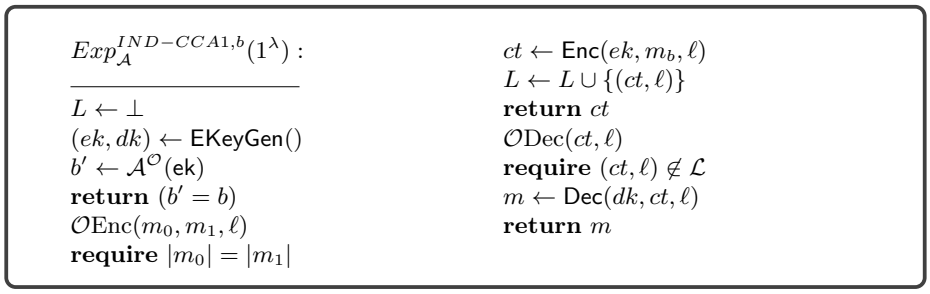


Fig. 5. Labelled IND-CCA1 game-based security definition.

Theorem 4 (smart meter data privacy). *Assuming the underlying encryption scheme is IND-CCA1 secure, the signature scheme is EUF-CMA secure, and the zero-knowledge proof system satisfies zero-knowledge proof of knowledge properties, our PPDR scheme achieves smart meter data privacy as defined in Definition 1.*

We define a a sequence of games starting from $Exp_{\mathcal{A},\mathcal{DR}}^{DRPRIV,0}$ and ending at $Exp_{\mathcal{A},\mathcal{DR}}^{DRPRIV,1}$. At each transition, we bound the distinguishing advantage of $\mathcal{A}$. This sequence allows us to derive an overall bound establishing the privacy guarantee of our DR scheme. We first initiate the PPDR scheme with the following algorithms ($\mathsf{Setup}, \mathsf{Report}, \mathsf{Eval}, \mathsf{Publish}, \mathsf{Verify}$), which are defined in $Exp_{\mathcal{A},\mathcal{DR}}^{DRPRIV,b}$.

The Setup algorithm establishes the public parameters and creates the encryption key pair (ek, dk). It generates a unique signing key pair (sk, pk) for each id, along with a proof of knowledge, and computes the encrypted reduction target list $\mathbf{ct_T}$.

The oracle $\mathcal{O}\mathrm{ReportIND}(id, m_0, m_1, t)$ uses the *report* function to encrypt the report plaintext m_b using the ElGamal encryption scheme with ek. It generates a proof of plaintext knowledge, signs the resulting ciphertext and proof using sk, and outputs the signed report $Rt_b = (pk, t, \mathbf{ct}_b, \boldsymbol{\pi}_{\boldsymbol{m_b}}, \sigma)$.

The oracle $\mathcal{O}\mathrm{View}()$ runs the publishing procedure to produce the anonymized list $(\mathbf{pk}', \pi_{mix})$. It also publishes the anonymized ciphertext tuples $(pk_i', \mathbf{ct_i}, \boldsymbol{\pi}_{\boldsymbol{m_i}}, t, \pi_i)$ for all participant. We assume that for the selected pk_i', the corresponding report ciphertext $\mathbf{ct}_i$ consists of two ciphertexts, representing the baseline ciphertext and the consumption ciphertext, respectively.

The function *eval* has two main computations. First, it uses the comparison protocol to determine whether the reported consumption is below the baseline value. Second, it computes the aggregate reduction over all reports and checks whether this value matches any element in the encrypted target list. The algorithm outputs the evaluation result R together with the corresponding proofs of correctness Π.

Proof. We now prove that our scheme preserves smart meter data privacy. We use the game-based technique to bound the adversary's advantage. Since oracles

can be invoked multiple times by the adversary, we will omit pre-factors in the advantage bounds.

Game G_0 : Let $\boldsymbol{G_0}$ be the initiation of the DRPRIV game corresponding to $Exp^{DRPRIV,0}_{\mathcal{A},\mathcal{DR}}$, where $\mathcal{A}$ view of the game is $\mathcal{B}_0$ and the evaluation result (R, Π_0) is computed exactly on $\mathcal{B}_0$. We denote the advantage of $\mathcal{A}$ in game $\boldsymbol{G_0}$ with $Adv^{\boldsymbol{G_0}}_{\mathcal{A}}$.

Game G'_0 : In $\boldsymbol{G'_0}$, the challenger simulates all NIZKP that they generates in game $\boldsymbol{G_0}$, namely all NIZKP of knowledge π_{sk_i}, π_{m_i}, π_{dk}, and the evaluation proof $\Pi = (\pi_{ord}, \boldsymbol{\pi_{dec}}, \boldsymbol{\pi_{eq}},$
$(\pi_i)_{i\in T})$. By the zero-knowledge property, $\mathcal{A}$ cannot extract any knowledge from the simulated proofs, and therefore we have: $Adv^{\boldsymbol{G_0}}_{\mathcal{A}} \leq Adv^{\boldsymbol{G'_0}}_{\mathcal{A}} + Adv^{\boldsymbol{zkp}}_{\mathcal{B}}$.

Note that the visible view $\mathcal{B}_0$ is generated by querying $\mathcal{O}\text{ReportIND}(id, m_0, m_1, t)$ and $\mathcal{O}\text{Append}(Rt_{\mathcal{A}})$. For each entry (id_i, Rt_i) in visible view $\mathcal{B}_0$ the challenger in reduction game associates the tuple $(id_i, m^0_i, m^1_i, Rt^0_i, Rt^1_i)$ where Rt^0_i, Rt^1_i are the ciphertext corresponding with m^0_i, m^1_i. The adversary view of the i-th entry in $\mathcal{B}_0$ is $Rt_i = Rt^0_i$. If the entry (id_i, Rt_i) is created by querying the oracle $\mathcal{O}\text{Append}(Rt_{\mathcal{A}})$ the associated tuple is $(id_i, \varnothing, \varnothing, Rt_i, Rt_i)$. Note that in $\mathcal{O}$Append, $\mathcal{A}$ cannot append a report ciphertext Rt on behalf of an honest identity, since the signature scheme is EUF-CMA secure.

Let $\mathcal{B}_0$ be the view of $\mathcal{A}$ in $\boldsymbol{G'_0}$. $\mathcal{B}_0$ contains n different report ciphertexts Rt^0_i corresponding to n different id. We define the following n games G'_i for $i = 1, \ldots, n$ by changing the content $\mathcal{B}_0$ in the game $\boldsymbol{G'_0}$. We denote by $\mathcal{B}^i_0$ the output view of the game $\boldsymbol{G'_i}$.

Game G'_i : This game is obtained from $\boldsymbol{G'_{i-1}}$ when the challenger replaces Rt^0_i i-th report ciphertext from $\mathcal{B}^0_{i-1}$ with the corresponding Rt^1_i.

We now show that if $\boldsymbol{G'_i} \neq \boldsymbol{G'_{i-1}}$, the advantage of $\mathcal{A}$ in distinguishing $\boldsymbol{G'_i}$ from $\boldsymbol{G'_{i-1}}$ is negligible. If $\boldsymbol{G'_i} = \boldsymbol{G'_{i-1}}$, the advantage of $\mathcal{A}$ does not change. We argue that our encryption scheme with the non-malleable proofs of plaintext knowledge satisfies $IND - CCA1$ security. Let $\mathcal{C}$ be an IND-CCA1 adversary who received the challenge ciphertext ct^* for the query (m_0, m_1). The adversary $\mathcal{C}$ uses the advantage of a DRPRIV adversary $\mathcal{A}$ in distinguishing the games $\boldsymbol{G'_i}$ and $\boldsymbol{G'_{i-1}}$. To this end, $\mathcal{C}$ simulates the games $\boldsymbol{G'_i}$ and $\boldsymbol{G'_{i-1}}$ to $\mathcal{A}$.

- for all $j < i$ compute $Rt^1_j \leftarrow \text{report}(id_j, sk, ek, m^1_j, t)$
- for all $j > i$ compute $Rt^0_j \leftarrow \text{report}(id_j, sk, ek, m^0_j, t)$
- for $j = i$ set $Rt_j = (pk_i, ct^*, \pi_m*, t, \sigma)$, where ct^* is the challenge ciphertext from the IND-CCA1 security game on query (m_0, m_1). Note the proof π_m* can be simulated by $\mathcal{C}$ according to $\boldsymbol{G'_0}$.
- $\mathcal{C}$ compute the evaluation result based on $(m^0_1, m^0_2, \ldots,$
  $m^0_{i-1}, m_0, m^0_{i+1}, \ldots m^0_n)$ and simulate all evaluation proofs for the visible view $\mathcal{B}^i_0$. In the evaluation part, $\mathcal{C}$ uses the decryption oracle of the $IND - CCA1$ security game to decrypt the adversary's ciphertexts, and includes them in the evaluation result.

Therefore, the distinguishing advantage of $\mathcal{A}$ between these transition games is at most $Adv^{\boldsymbol{IND-CCA1}}_{\mathcal{C}}$.

Game G_1 : Let $\boldsymbol{G_1} = \boldsymbol{G'_n}$ be the n-th transition game, where the view of the $\mathcal{A}$ is equal to the view of $\mathcal{A}$ in the $Exp^{DRPRIV,1}_{\mathcal{A},\mathcal{DR}}$. We can conclude that the advantage of the adversary through all transitions from $\boldsymbol{G_0}, \boldsymbol{G'_0}, \boldsymbol{G'_1}, \ldots, \boldsymbol{G'_n}, \boldsymbol{G_1}$ where $\boldsymbol{G_0} = Exp^{DRPRIV,0}_{\mathcal{A},\mathcal{DR}}$ and $\boldsymbol{G_1} = Exp^{DRPRIV,1}_{\mathcal{A},\mathcal{DR}}$ is bounded as follows:

$$Adv^{\boldsymbol{DRPRIV}}_{\mathcal{A}} \leq Adv^{\boldsymbol{zkp}}_{\mathcal{B}} + Adv^{\boldsymbol{EUF-CMA}}_{\mathcal{D}} + Adv^{\boldsymbol{IND-CCA1}}_{\mathcal{C}}$$

A.2 Smart Meter Participation Privacy

A DR scheme provides participation privacy by revealing only the total number of participants and the evaluation result. Our PPDR scheme captures participation privacy by anonymizing the public keys and the energy reports of participating smart meters. This anonymization is performed by a trusted entity, the EA, ensuring that the identity of a participating smart meter is indistinguishable from that of a non-participating one. In addition, we assume that each smart meter communicates with the EA over a secure channel, which prevents the adversary from linking an anonymized report to the identity of its sender.

Given a PPT adversary $\mathcal{A}'$, we define the experiment $Exp^{DRPAR,b}_{\mathcal{A},\mathcal{DR}}$, which models an indistinguishability game involving two views that are maintained simultaneously. Only one of these views is revealed to $\mathcal{A}'$, depending on the bit $b \in \{0,1\}$ of $Exp^{DRPAR,b}_{\mathcal{A},\mathcal{DR}}$, used in the Figure 6.

The challenger flips a coin and executes the setup phase of the DR scheme. The adversary $\mathcal{A}'$ may issue multiple queries to the oracle $\mathcal{O}$Report which allows an honest smart meter id to generate a ciphertext report for a given plaintext report m at timestamp t in both views $\mathcal{B}_0$ and $\mathcal{B}_1$. Additionallg, $\mathcal{A}'$ can call the oracle $\mathcal{O}$Append to append a report on behalf of any smart meter. By using these oracles, $\mathcal{A}'$ populates both views with additional content so that both $\mathcal{B}_0$ and $\mathcal{B}_1$ contain the same reports, except for the smart meters with id_0 and id_1. To model participation versus non-participation, we provide an oracle $\mathcal{O}$Participate that, in $\mathcal{B}_0$ allows the honest smart meter id_0 to participate in the DR program by reporting m, while id_1 does not participate (and vice versa in $\mathcal{B}_1$). The adversary may query $\mathcal{O}Eval$ to obtain the result R and the corresponding proof Π_b for view $\mathcal{B}_b$.

Definition 3. *Consider a DR scheme for a set $\mathbb{ID}$ of identities and with the DR parameters $\mathbb{DR}$. We say that the DR scheme achieves participation privacy if, for all PPT adversaries $\mathcal{A}'$,*

$$\left| Pr\left[Exp^{DRPAR,0}_{\mathcal{A}',DR}(1^\lambda) = 1\right] - Pr\left[Exp^{DRPAR,1}_{\mathcal{A}',DR}(1^\lambda) = 1\right] \right|$$

is negligible in the security parameter λ.

Theorem 5 (participation privacy.). *Our PPDR scheme achieves participation privacy according to Definition 3 if the signature is EUF-CMA and assuming the zero knowledge proof of knowledge properties.*

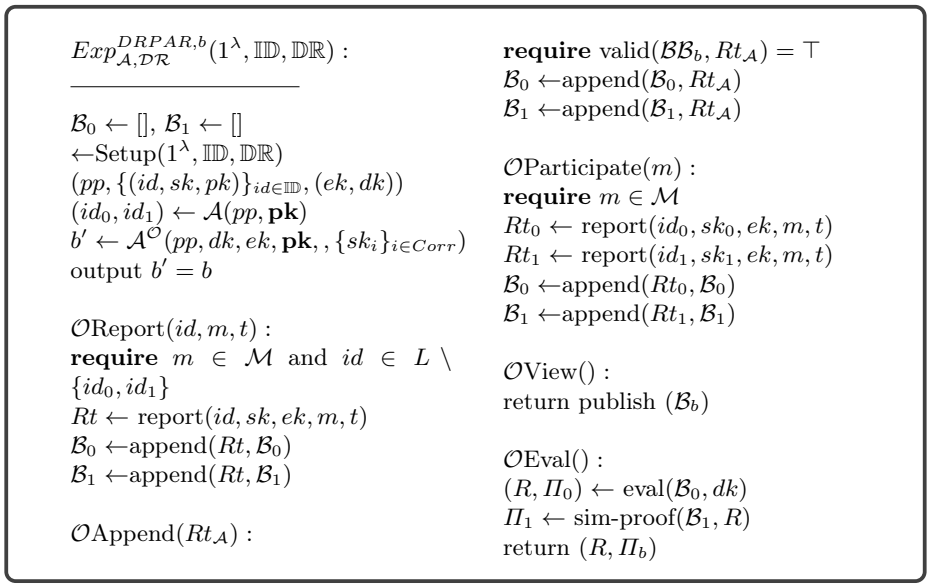


Fig. 6. The game-based security definition of participation privacy $Exp^{DRPAR,b}_{\mathcal{A},\mathcal{DR}}(1^\lambda, \mathbb{ID}, \mathbb{DR})$. The adversary wins if it guesses the bit b.

Proof. **Game G_0**. Let G_0 denote the initiation of our PPDR scheme corresponding to $\text{Exp}^{DRPAR,0}_{\mathcal{A},\mathcal{DR}}$. In this experiment, smart meter id_0 participates in the DR program while id_1 does not. The view of the adversary $\mathcal{A}'$ is $\mathcal{B}_0$, and the result (R, Π) is returned by the oracle $\mathcal{O}$Eval() computed on $\mathcal{B}_0$. In the PPDR scheme, each smart meter submits two reports: a baseline report and a consumption report. Without loss of generality, we assume that Rt contains all reports of a smart meter.

Game G'_0. Game G'_0 is obtained from G_0 by allowing the challenger, who holds the simulation trapdoor generated during setup, to simulate all zero-knowledge proofs. Since the zero-knowledge proof system ensures that the simulated proofs are computationally indistinguishable from those produced in G_0, G'_0 is indistinguishable from G_0. Moreover, in the oracle $\mathcal{O}$Append, $\mathcal{A}'$ cannot generate and submit a report on behalf of honest smart meters since the signature scheme is existentially unforgeable. We can conclude that the advantage of $\mathcal{A}'$ in distiguishing these two games is $Adv^{G'_0}_{\mathcal{A}'} \leq Adv^{zkp}_{\mathcal{B}} + Adv^{EUF-CMA}_{\mathcal{D}}$, where we denote the advantage of the adversary $\mathcal{B}$ in distinguishing simulations from real proofs with $Adv^{zkp}_{\mathcal{B}}$.

Game G''_0. Game G''_0 is derived from G'_0 by swapping the ciphertext components $\mathbf{ct_0}$ and $\mathbf{ct_1}$ returned by $\mathcal{O}$Participant(m). As a result,$(pk'_0, \mathbf{ct_1}, \boldsymbol{\pi}_{\boldsymbol{m_0}}, t, \pi_0)$ is placed on $\mathcal{B}_0$ and $(pk'_1, \mathbf{ct_0}, \boldsymbol{\pi}_{\boldsymbol{m_1}}, t, \pi_1)$ on $\mathcal{B}_1$. The related proofs $\boldsymbol{\pi}_{\boldsymbol{m_0}}$ and π_0 for this modification can be simulated according to G'_0. The distribution of $\mathbf{ct_0}$ is exactly the same as the distribution of $\mathbf{ct_1}$. Because in both games, the cipher-

text is an honest encryption of the plaintext m with fresh randomness sampled from the same distribution. Therefore the two games $\boldsymbol{G''_0}$ and $\boldsymbol{G'_0}$ are identical in distribution, and the adversary's advantage is equal: $Adv_{\mathcal{A}'}^{\boldsymbol{G''_0}} = Adv_{\mathcal{A}'}^{\boldsymbol{G'_0}}$.

Game $\boldsymbol{G_1}$. $\boldsymbol{G_1}$ is obtained from $\boldsymbol{G''_0}$ by swapping the anonymized public keys pk'_0 and pk'_1. As a result, $Rt'_1 = (pk'_1, \mathbf{ct_1}, \boldsymbol{\pi_{m_1}}, t, \pi_1)$ is placed on $\mathcal{B}_0$, instead of $Rt'_0 = (pk'_0, \mathbf{ct_1}, \boldsymbol{\pi_{m_0}}, t, \pi_0)$. The advantage of $\mathcal{A}'$ in distinguishing $\boldsymbol{G''_0}$ and $\boldsymbol{G_1}$ is as follows.

Let us consider the tuple $(pk_0, pk_1, pk'_b, pk'_{1-b})$ where (pk_0, pk_1) are the public keys of (id_0, id_1), and (pk'_b, pk'_{1-b}) are the corresponding anonymized public keys. We have two different versions of this tuple:

- $(pk_0, pk_1, pk'_0, pk'_1)$, which corresponds to the game $\boldsymbol{G''_0}$ with $b = 0$ where pk'_0 is participating.
- $(pk_0, pk_1, pk'_1, pk'_0)$, which corresponds to the game $\boldsymbol{G_1}$ with $b = 1$ where pk'_1 is participating.

Since $pk'_0 = pk_0^{r_0}$ and $pk'_1 = pk_1^{r_1}$ with uniform randomness $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q$, both games generate tuples that have same distribution, namely, $(pk_0, pk_1, pk'_0, pk'_1) \stackrel{d}{=} (pk_0, pk_1, pk'_1, pk'_0)$. This means that the advantage of $\mathcal{A}'$ in distinguishing $\boldsymbol{G''_0}$ from $\boldsymbol{G_1}$ is information-theoretically hard. Hence , we have: $Adv_{\mathcal{A}'}^{\boldsymbol{G''_0}} = Adv_{\mathcal{A}'}^{\boldsymbol{G_1}}$.

Finally, we argue that through the above transition games we have changed the view of $\mathcal{A}'$ from $\mathcal{B}_0$ in game $\boldsymbol{G_0}$ to $\mathcal{B}_1$ in game $\boldsymbol{G_1}$. We conclude that

$$Adv_{\mathcal{A}'}^{\boldsymbol{DRPAR}} \leq Adv_{\mathcal{B}}^{\boldsymbol{zkp}} + Adv_{\mathcal{D}}^{\boldsymbol{EUF-CMA}}$$

We assume that the anonymization values g^{r_0} and g^{r_1} are transmitted to the users (smart meters) over secure channels. However, even if these values are published on $\mathcal{BB}$ or otherwise revealed to $\mathcal{A}'$, the adversary's advantage in breaking participation privacy can be reduced to the advantage of an adversary $\mathcal{C}$ in solving the Decisional Diffie–Hellman (DDH) problem as follows. Let us consider an adversary $\mathcal{A}$ that, given a tuple (g^{r_0}, pk_0, pk'_b) , can distinguish whether $pk'_b = pk_0^{r_0}$ or not with non-negligible advantage. We construct a PPT algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to distinguish whether the given tuple (g, g^a, g^b, g^c) is a valid DDH tuple or a random one. The algorithm $\mathcal{B}$ receives $(g, X = g^a, Y = g^b, Z)$, and sets $(g^{r_0}, pk_0, pk'_b) = (X, Y, Z)$ to use the advantage of $\mathcal{A}$. If the output of $\mathcal{A}$ links pk'_b to (pk_0, g^{r_0}), then $\mathcal{B}$ outputs DDH; otherwise, it outputs random. Thus, any non-negligible advantage of $\mathcal{A}$ implies a non-negligible advantage of $\mathcal{B}$ against the DDH problem.

A.3 Verifiability

The verifiability of our PPDR scheme has the following properties. First, the scheme provides public verifiability: anyone can verify the correctness of both the anonymization procedure and the output of the evaluation in the activation phase. This ensures that the published result corresponds exactly to the value obtained by decrypting all reports on the $\mathcal{BB}$ and applying the evaluation function

$Exp^{DRVER}_{\mathcal{A},\mathcal{DR}}(1^\lambda, \mathbb{ID}, \mathbb{DR})$:
$(pp, ek, dk) \leftarrow$Setup(λ)
$H, C \leftarrow []$
$(\mathcal{B}, R, \Pi) \leftarrow \mathcal{A}^{\mathcal{O}}(c_n, h_n, ek, dk, \mathbb{ID})$
if $R = \perp$ or $\mathsf{Verify}(\mathcal{B}, R, \Pi) = \perp$
then return $\perp$
end if
if $\exists\{(id_{h'_1}, m_{h'_1}, *), \ldots, (id_{h'_n}, m_{h'_n}, *)\} \subseteq H$
and $\exists\{m_{c'_1}, \ldots, m_{c'_n}\}$ s.t.
$R = f(m_{h'_1}, \ldots, m_{h'_n}, m_{c'_1}, \ldots, m_{c'_n})$
then
return $\perp$
else
return $\top$
end if

$\mathcal{O}$Register(id) :
require $id \notin \mathbb{U}$
$\mathbb{U} = \mathbb{U} \cup (id, pk, sk)$
return $\mathbb{ID} \cup (id, pk)$

$\mathcal{O}$Corrupt(id) :
require $id \in \mathbb{ID}$ and $(id, *) \notin H$
return $C = C \cup (id, pk, sk)$

$\mathcal{O}$Append(id, Rt) :
require $\mathsf{Validate}(\mathcal{B}, Rt) = \top$
$\mathcal{B} \leftarrow$append$(Rt, \mathcal{B})$

$\mathcal{O}$Report(id, m, t) :
require $m \in \mathcal{M}$ and $id \in H$
$Rt \leftarrow$ report(id, sk, ek, m, t)
$\mathcal{B} \leftarrow$append$(Rt, \mathcal{B})$
$H \leftarrow$append$((id, m, Rt), H)$

Fig. 7. The game based definition of verifiability $Exp^{ver}_{DR,\mathcal{A}}(\lambda)$, where function f computes the result R on plaintexts.

to the resulting plaintexts. Furthermore, anyone can verify that the $\mathcal{BB}$ contains only reports originating from registered SMs, and that each smart meter identity contributes at most two ciphertexts (a baseline and an energy consumption report). We assume that the adversary $\mathcal{A}$ controls a subset of SMs, as well as EA and DRA. In particular, if a user verifies the $\mathcal{BB}$ after their participation or after submitting their energy report, our scheme guarantees verifiability even against a malicious $\mathcal{BB}$. The user can detect any attempt to omit, alter, or replace their report.

We aim to show that the adversary has only a negligible probability of producing a $\mathcal{BB}$ and an evaluation result that violate the following correctness properties: 1) All selected honest users who verify their reports are included in the evaluation. 2) For honest users who do not verify their reports, the adversary may delete their contributions, but cannot modify them. 3) For each corrupted user, at most two reports may be included.

Let $\mathbb{ID}$ denote the list of registered SMs public keys, and let $\mathbb{U}$ denote the list of SMs secret–public key pairs. We denote by C and H the sets of corrupted and honest SMs, respectively. The set H contains the tuples of form (id, m, Rt). The honest SMs do not leak the signing keys, and their users can check their participation in the DR program and their energy report Rt on $\mathcal{BB}$. The adversary $\mathcal{A}$ interacts with the following oracles, which form the experiment Exp^{ver} defined in Figure 7.

– $\mathcal{O}$register(id), which allows $\mathcal{A}$ to get a list of registered SMs. It generates (id, sk, pk), returns (id, pk), and updates $\mathbb{ID} \cup (id, pk)$.

- $\mathcal{O}$corrupt(id), which allows $\mathcal{A}$ to obtain (id, sk, pk) given a registered smart meter id. It checks that $(id, *) \in \mathbb{ID}$ and $(id, *) \notin H$, then returns (id, sk, pk) and updates $C \cup (id, sk, pk)$.
- $\mathcal{O}$Append(id, Rt), which allows $\mathcal{A}$ to append a valid report Rt to the $\mathcal{BB}$.
- $\mathcal{O}$Report(id, m), which allows $\mathcal{A}$ to get a valid report Rt corresponding the honest SM $id \in H$.

Definition 4 (DRVER). *Consider a DR scheme for a set $\mathbb{ID}$ of identities and DR parameters $\mathbb{DR}$. We say that the DR scheme achieves verifiability if, for every PPT adversary $\mathcal{A}$,*

$$\Pr\big[\, Exp^{\mathsf{DRVER}}_{\mathcal{A},\mathcal{DR}}(1^\lambda, \mathbb{ID}, \mathbb{DR}) = 1 \,\big]$$

is negligible in the security parameter λ.

Theorem 6 (verifiability.). *Our PPDR scheme achieves public verifiability according to Definition 4 if the signature is EUF-CMA and assuming the soundness of the zero knowledge proof of knowledge, and the correctness of the encryption scheme.*

Proof. Assume the adversary outputs a bulletin board $\mathcal{B}$ and an evaluation result (R, Π) that pass all verification checks. Public verifiability ensures that all posted reports are valid: the well-formedness proof $\pi_{\mathbf{m}}$ guarantees message-space correctness; the soundness of the zk-SNARK proof of signature knowledge π ensures correct ciphertext anonymization; the soundness of the mixing proof π_{mix} guarantees correctness of the public-key mixing; and finally, the soundness of the evaluation proof $\Pi = (\pi_{ord}, \boldsymbol{\pi_{dec}}, \boldsymbol{\pi_{eq}}, (\pi_i)_{i \in T})$, consisting of the ordering proof π_{ord}, the decryption proofs $\boldsymbol{\pi_{dec}}$ and $(\pi_i)_{i \in T}$, the plaintext-equality test proofs $\boldsymbol{\pi_{eq}}$ together with the correctness of the homomorphic encryption scheme, ensure the correctness of the final evaluation result.

Let us consider $id \in H$. Since all honest reports are signed using EUF-CMA secure signatures, and the NIZKP of signature knowledge is sound, $\mathcal{A}$ cannot forge a signature, nor can it produce a valid proof on a ciphertext belonging to an honest SM identity. Thus, honest ciphertext reports posted on $\mathcal{B}$ cannot be modified; they may only be deleted. Consequently, for every honest SM $id \in H$ that verifies its participation on $\mathcal{B}$, the associated report ciphertext Rt must be included in the evaluation without alteration or replacement. For the remaining $id \in H$ who do not verify their participation in the DR program or their energy reports, $\mathcal{A}$ may choose to delete their reports, but cannot modify them.

Next, let us suppose that a valid report ciphertext on $\mathcal{B}$ is associated with an identity not in H. Such a ciphertext must originate from a corrupted SM; otherwise, $\mathcal{A}$ would win the game. Since the signature scheme is EUF-CMA secure, $\mathcal{A}$ cannot forge a valid signature on behalf of an honest identity. Moreover, by the soundness of the NIZKP of signature knowledge, any ciphertext accompanied by a valid proof must correspond to a genuine signature under some registered secret key. Therefore, $\mathcal{A}$ cannot generate a valid report ciphertext without knowing the corresponding secret signing key. Hence, the report ciphertext must belong

to a corrupted SM $id \in C$. In our DR scheme, each selected SM submits at most two report ciphertexts. The adversary $\mathcal{A}$ can generate at most two valid ciphertexts for each corrupted SM and cannot exceed this bound. We conclude that the evaluation result R must be computed over a subset of report ciphertexts of honest SMs H, together with at most $|C|$ valid report ciphertexts originating from corrupted users.

B Instantiations

Schnorr signature scheme. Let p, q be primes with $q \mid (p-1)$ and let $G \subset \mathbb{F}_p^*$ be the cyclic subgroup of order q with generator g. Let the message space be $\mathcal{M} \subseteq \{0,1\}^*$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle (with an explicit domain separator $\mathsf{dom}_{\mathsf{sig}}$).

A signature scheme $\mathsf{Sig}_{\mathsf{Schnorr}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is defined as follows:

- $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: sample $x \xleftarrow{\$} \mathbb{Z}_q$ and set $pk \leftarrow g^x \in G$. Output $sk \leftarrow x$ and pk.
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: on input x and $m \in \mathcal{M}$, sample $t \xleftarrow{\$} \mathbb{Z}_q$ and compute $\ell \leftarrow g^t$. Let $e \leftarrow H(\mathsf{dom}_{\mathsf{sig}} \parallel m \parallel \ell \parallel pk) \in \mathbb{Z}_q$ and $s \leftarrow t + e \cdot x \bmod q$. Output the signature $\sigma \leftarrow (\ell, s)$.
- $b \leftarrow \mathsf{Verify}(pk, m, \sigma)$: on input pk, m, and $\sigma = (\ell, s)$, compute $e \leftarrow H(\mathsf{dom}_{\mathsf{sig}} \parallel m \parallel \ell \parallel pk)$ and output $b = 1$ iff $g^s = \ell \cdot pk^e$ in G (otherwise output $b = 0$).

Security. Under the discrete logarithm assumption in G, the scheme $\mathsf{Sig}_{\mathsf{Schnorr}}$ is EUF–CMA secure in the random-oracle model.

Proof of private key knowledge. We instantiate Proof_{sk} and Verify_{sk} with the Schnorr Σ-protocol for discrete logarithms over the group G and generator g used by $\mathsf{Sig}_{\mathsf{Schnorr}}$, transformed into a NIZK via Fiat–Shamir. Let $H_{\mathsf{sk}} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle with an explicit domain separator $\mathsf{dom}_{\mathsf{sk}}$. For a public statement $x_{sk} = (id, pk)$ and witness $\omega_{sk} = sk$ such that $pk = g^{sk}$, we define:

- $\pi_{sk} \leftarrow \mathsf{Proof}_{sk}(x_{sk}, \omega_{sk})$: on input public statement $x_{sk} = (id, pk)$ and witness $\omega_{sk} = sk$,
  1. sample $r \xleftarrow{\$} \mathbb{Z}_q$ and compute $\alpha \leftarrow g^r$;
  2. compute the challenge
  $$e \leftarrow H_{\mathsf{sk}}\big(\mathsf{dom}_{\mathsf{sk}} \parallel id \parallel pk \parallel \alpha\big) \in \mathbb{Z}_q;$$
  3. compute the response $z \leftarrow r + e \cdot sk \bmod q$;
  4. output the proof $\pi_{sk} \leftarrow (\alpha, z)$.
- $b \leftarrow \mathsf{Verify}_{sk}(x_{sk}, \pi_{sk})$: on input the public statement $x_{sk} = (id, pk)$ and a proof $\pi_{sk} = (\alpha, z)$,
  1. recompute
  $$e \leftarrow H_{\mathsf{sk}}\big(\mathsf{dom}_{\mathsf{sk}} \parallel id \parallel pk \parallel \alpha\big) \in \mathbb{Z}_q;$$

2. output $b = 1$ iff
$$g^z = \alpha \cdot pk^e \quad \text{in } G,$$
and $b = 0$ otherwise.

Proof of correct decryption. We instantiate $(\mathsf{Proof}_{dk}, \mathsf{Verify}_{dk})$ as a non-interactive zero-knowledge proof of knowledge (NIZKPoK) of an equality of DLogs in the style of Chaum–Pedersen [5].

Recall the exponential ElGamal scheme $\mathsf{AHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ over a group G of order q with generator g. For convenience, we write the public encryption key as
$$ek = (p, q, g, pk) \quad \text{with} \quad pk = g^{dk} \in G$$
and a ciphertext as $ct = (C_1, C_2)$ with
$$C_1 = g^r, \qquad C_2 = g^m \cdot pk^r$$
for some $r \xleftarrow{\$} \mathbb{Z}_q$. Given ek, a message m and a ciphertext ct, we define the auxiliary value
$$M := g^m, \qquad D := C_2/M.$$
By correctness of encryption, we have
$$D = \frac{C_2}{g^m} = \frac{g^m \cdot pk^r}{g^m} = pk^r = (g^{dk})^r = C_1^{dk},$$
and therefore
$$\log_g(pk) = dk = \log_{C_1}(D).$$
We can thus rewrite the decryption relation R_{dk} equivalently as
$$\big((ek, m, ct), dk\big) \in R_{dk} \quad \text{iff} \quad ek = (p, q, g, pk),\ ct = (C_1, C_2)$$
and
$$pk = g^{dk} \ \wedge \ D = C_1^{dk},$$
where $D = C_2/g^m$ is uniquely determined by (ek, m, ct). This is exactly an *equality-of-discrete-logs* statement:
$$\log_g(pk) = \log_{C_1}(D).$$

Chaum–Pedersen protocol. On input the public statement $x_{dk} = (ek, m, ct)$ and witness $\omega_{dk} = dk$, the prover and verifier first derive
$$(g, pk, C_1, C_2) \leftarrow ek, ct, \qquad M := g^m, \quad D := C_2/M.$$
The underlying Chaum–Pedersen protocol (in its interactive form) is:

– *Prover's commitment.* Sample $t \xleftarrow{\$} \mathbb{Z}_q$ and compute
$$T_1 := g^t, \qquad T_2 := C_1^t.$$
Send (T_1, T_2) to the verifier.

- *Challenge.* The verifier samples $c \xleftarrow{\$} \mathbb{Z}_q$ (or, in the NIZK version, c will be derived via Fiat–Shamir) and sends c to the prover.
- *Response.* The prover computes

$$s := t + c \cdot dk \bmod q$$

  and sends s to the verifier.
- *Verification.* The verifier recomputes $M := g^m$ and $D := C_2/M$, and checks

$$g^s \stackrel{?}{=} T_1 \cdot pk^c \qquad \text{and} \qquad C_1^s \stackrel{?}{=} T_2 \cdot D^c.$$

  The verifier outputs $b = 1$ if both equations hold, and $b = 0$ otherwise.

To obtain a NIZKPoK in the random oracle model, we apply the Fiat–Shamir transform and derive the challenge as

$$c := H(ek, m, ct, T_1, T_2),$$

for a hash function H modeled as a random oracle. The resulting non-interactive proof is

$$\pi_{dk} := (T_1, T_2, s),$$

and the algorithms $(\mathsf{Proof}_{dk}, \mathsf{Verify}_{dk})$ are obtained by running the prover and verifier steps above with c computed via H. By construction, any accepting proof (x_{dk}, π_{dk}) certifies that there exists a secret key dk such that $pk = g^{dk}$ and ct is a valid encryption of m under ek, i.e., $((ek, m, ct), dk) \in R_{dk}$.

Proof of knowledge of r for *EPET*. We instantiate Proof_r and Verify_r with a Schnorr-type proof of knowledge of a discrete logarithm over the ciphertext group. Let C denote the ciphertext group with component-wise operation and order q, and let $H_r : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle with an explicit domain separator dom_r. For a public statement $x_r = (ct_1, ct_2, ct_{eq})$ and a witness $\omega_r = r \in \mathbb{Z}_q$ such that $ct_{eq} = (ct_1/ct_2)^r$ in C, we define:

- $\pi_r \leftarrow \mathsf{Proof}_r(x_r, \omega_r)$: on input $x_r = (ct_1, ct_2, ct_{eq})$ and $\omega_r = r$,
  1. compute the base $B \leftarrow ct_1/ct_2$ in C;
  2. sample $t \xleftarrow{\$} \mathbb{Z}_q$ and compute the commitment $T \leftarrow B^t$;
  3. compute the challenge

$$e \leftarrow H_r\big(\mathsf{dom}_r \parallel ct_1 \parallel ct_2 \parallel ct_{eq} \parallel T\big) \in \mathbb{Z}_q;$$

  4. compute the response $z \leftarrow t + e \cdot r \bmod q$;
  5. output the proof $\pi_r \leftarrow (T, z)$.
- $b \leftarrow \mathsf{Verify}_r(x_r, \pi_r)$: on input the public statement $x_r = (ct_1, ct_2, ct_{eq})$ and a proof $\pi_r = (T, z)$,
  1. recompute $B \leftarrow ct_1/ct_2$ in C;
  2. recompute

$$e \leftarrow H_r\big(\mathsf{dom}_r \parallel ct_1 \parallel ct_2 \parallel ct_{eq} \parallel T\big) \in \mathbb{Z}_q;$$

Algorithm 1 CGate for two aggregators

Require: G, a group of prime order q with public generator g; an exponential ElGamal encryption key ek with corresponding decryption key dk; a public element $\tilde{h} \in G$ independent of ek; $C = (c_1, c_2)$, an encryption of some $m \in \{0,1\}$ under ek; $C' = (c'_1, c'_2)$, an encryption of $m' \in \{0,1\}$ under ek.

Ensure: C'', a random encryption of $m \cdot m'$ under ek, and a verification transcript Π_{CGate}.

1: $C'_0 \leftarrow ({c'_1}^{-1}, {c'_2}^{-1}) = C'^{-1}$ ▷ component-wise inverse encryption of C'
2: $C_0 \leftarrow C$
3: **for** $i = 1$ **to** 2 **(for aggregator** i**) do**
4: $(u, v) \leftarrow C_{i-1}$
5: $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$; $s \xleftarrow{\$} \{-1, 1\}$
6: $C_i \leftarrow \left(u^s g^{r_1},\ v^s ek^{r_1}\right)$; $e \leftarrow \tilde{h}^{r_1}$
7: $C'_i \leftarrow \mathsf{ReEnc}_{ek}({C'_{i-1}}^{s}, r_2)$ ▷ if $i = 1$
$\mathsf{ReEnc}_{ek}({C'_{i-1}}^{s}, r_2) = (g^{r_2}.C'_0[1]^s, ek^{r_2}.C'_0[2]^s)$
8: **broadcast** C_i, e, C'_i and a ZKP π_i that they are well formed
9: Let $\Pi \leftarrow (C_2, C'_2, \pi_2) \,\|\, (C_1, C'_1, \pi_1)$
10: Each aggregator verifies the proofs π_1, π_2 of the other aggregator
11: The aggregators jointly rerandomize C_2 and C'_2 into X' and Y', obtaining a transcript Π_{ReEnc}
12: The aggregators jointly decrypt Y' using dk to obtain $m_2 \in \{0,1\}$, producing a decryption transcript Π_{Dec}
13: **Output** a ciphertext

$$C'' \leftarrow C \cdot (X')^{m_2} =$$

(with multiplication and exponentiation applied component-wise), and set

$$\Pi_{\mathsf{CGate}} \leftarrow (m_2, \Pi_{\mathsf{Dec}}) \,\|\, (X', Y', \Pi_{\mathsf{ReEnc}}) \,\|\, \Pi.$$

3. output $b = 1$ iff

$$B^z = T \cdot ct_{eq}^{e} \quad \text{in } C,$$

and $b = 0$ otherwise.

Under the discrete logarithm assumption in the ciphertext group C, this instantiation yields a NIZK proof of knowledge of r for the relation R_r in the random-oracle model.

Proof of each aggregator.

$$\mathcal{R}_{\text{well}} = \Big\{ \big((C_{i-1}, C_i, C'_{i-1}, C'_i, e)\,;\,(s, r_1, r_2)\big) \;:\; C_i = \left(u^s g^{r_1},\ v^s ek^{r_1}\right) \wedge\ C'_i = \mathsf{ReEnc}_{ek}(C_{i-1}^{s}, r_2) \wedge\ e = \tilde{h}^{\,r_1} \Big\}$$

Proof of re-encryption.

An ElGamal ciphertext of a message $g^m \in G$ under public key $ek = g^d k$ is

$$C = (C_1, C_2) = (g^r,\ m \cdot (ek)^r)$$

Algorithm 2 PoK-CGate in terms of (C_i, C'_i) and ek

Require: A cyclic group G of prime order q with generator g; an exponential ElGamal key pair (ek, dk) over G; ciphertexts $C_i, C'_i, C_{i-1}, C'_{i-1}$ and an element $e \in G$; witnesses $r_1, r_2 \in \mathbb{Z}_q$ and $s \in \{-1, 1\}$ such that

$$C_i = \mathsf{ReEnc}_{ek}(C_{i-1}^{s}, r_1), \quad C'_i = \mathsf{ReEnc}_{ek}({C'_{i-1}}^{s}, r_2), \quad e = \tilde{h}^{\,r_1}.$$

1: Sample $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$
2: Compute

$$c_{s,C} \leftarrow \mathsf{Enc}_{ek}(0; \alpha), \quad c_{s,C'} \leftarrow \mathsf{Enc}_{ek}(0; \beta), \quad c_{s,e} \leftarrow \tilde{h}^{\,\alpha}.$$

3: Sample $d_{-s}, a_{-s,C}, a_{-s,C'} \xleftarrow{\$} \mathbb{Z}_q$
4: Using d_{-s} and $a_{-s,\cdot}$, form the simulated branch:

$$c_{-s,C} \leftarrow \mathsf{Enc}_{ek}(0; a_{-s,C}) \cdot \left(C_i C_{i-1}^{s}\right)^{-d_{-s}},$$

$$c_{-s,C'} \leftarrow \mathsf{Enc}_{ek}(0; a_{-s,C'}) \cdot \left(C'_i {C'_{i-1}}^{s}\right)^{-d_{-s}},$$

$$c_{-s,e} \leftarrow \tilde{h}^{\,a_{-s,C}} \cdot e^{-d_{-s}}.$$

5: Compute the Fiat–Shamir challenge

$$d \leftarrow \mathsf{hash}\big(ek, C_{i-1}, C'_{i-1}, C_i, C'_i, c_{+1,C}, c_{+1,C'}, c_{-1,C}, c_{-1,C'}, c_{+1,e}, c_{-1,e}\big).$$

6: Derive the challenge for the real branch:

$$d_s \leftarrow d - d_{-s}.$$

7: Set the responses for the real branch:

$$a_{s,C} \leftarrow \alpha + r_1 d_s, \quad a_{s,C'} \leftarrow \beta + r_2 d_s.$$

8: **Output** the proof

$$\pi_i = \big(c_{+1,C}, c_{+1,C'}, c_{-1,C}, c_{-1,C'}, c_{+1,e}, c_{-1,e}, d_{+1}, d_{-1}, a_{+1,C}, a_{+1,C'}, a_{-1,C}, a_{-1,C'}\big).$$

A re-encryption of C using fresh randomness $r' \in \mathbb{Z}_q$ is

$$C' = (C'_1, C'_2) = (C_1 g^{r'}, C_2 ek^{r'}) = (g^{r+r'},\ m \cdot ek^{r+r'})$$

To prove that C' is a correct re-encryption of C, compute $C''_1 = \frac{C'_1}{C_1}$ and $C''_2 = \frac{C'_2}{C_2}$ then prove

$$\log_g(C''_1) = \log_{ek}(C''_2)$$

This equality can be proven using a zero-knowledge Sigma protocol a Chaum–Pedersen proof.

Algorithm 3 Ver-CGate in terms of (C_i, C'_i) and ek

Require: A group G of prime order q; an exponential ElGamal public key ek; ciphertexts $C_i, C'_i, C_{i-1}, C'_{i-1}$ and an element $e \in G$; a proof

$$\pi = (c_{1,C}, c_{1,C'}, c_{-1,C}, c_{-1,C'}, c_{1,e}, c_{-1,e}, d_1, d_{-1}, a_{1,C}, a_{1,C'}, a_{-1,C}, a_{-1,C'}).$$

1: **Reject** if C_i has the form $(1_G, *)$ ▷ exclude trivial/degenerate ciphertexts
2: Compute the Fiat–Shamir challenge

$$d \leftarrow \mathsf{hash}\big(ek, C_{i-1}, C'_{i-1}, C_i, C'_i, c_{1,C}, c_{1,C'}, c_{-1,C}, c_{-1,C'}, c_{1,e}, c_{-1,e}\big).$$

3: Check the following equalities:
4: $d_1 + d_{-1} \stackrel{?}{=} d$;
5: $\mathsf{Enc}_{ek}\big(0; a_{1,C}\big)\,\big(C_i/C_{i-1}\big)^{-d_1} \stackrel{?}{=} c_{1,C}$;
6: $\mathsf{Enc}_{ek}\big(0; a_{1,C'}\big)\,\big(C'_i/C'_{i-1}\big)^{-d_1} \stackrel{?}{=} c_{1,C'}$;
7: $\tilde{h}^{\,a_{1,C}}\, e^{-d_1} \stackrel{?}{=} c_{1,e}$;
8: $\tilde{h}^{\,a_{-1,C}}\, e^{-d_{-1}} \stackrel{?}{=} c_{-1,e}$;
9: $\mathsf{Enc}_{ek}\big(0; a_{-1,C}\big)\,\big(C_i C_{i-1}\big)^{-d_{-1}} \stackrel{?}{=} c_{-1,C}$;
10: $\mathsf{Enc}_{ek}\big(0; a_{-1,C'}\big)\,\big(C'_i C'_{i-1}\big)^{-d_{-1}} \stackrel{?}{=} c_{-1,C'}$.
11: **If** all equalities hold, **return** 1; **otherwise return** 0.

Algorithm 4 : Protocol Overview

Setup and Key Generation

$pp \leftarrow \mathsf{Setup}(1^\lambda, \mathbb{ID}, \mathbb{DR})$

$\mathsf{EKeyGen}(pp)$:
$(ek, dk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\pi_{dk} \leftarrow \mathsf{Proof}_{dk}((pp, ek), dk)$
$ek \leftarrow (pp, ek, \pi_{dk})$
Output (ek, dk)

$\mathsf{SKeyGen}(id, pp)$:
For each $id \in \mathbb{ID}$:
$(sk, pk) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, id)$
$\pi_{sk} \leftarrow \mathsf{Proof}_{sk}((id, pp, pk), sk)$
$pk \leftarrow (pp, pk, \pi_{sk})$
Output $((id, pk), sk)$

ID Anonymization

$\mathsf{Anon}_{ID}(\mathbf{pk})$:
$\mathbf{pk}' \leftarrow \mathsf{Mix}(\mathbf{pk}, \mathbf{r}', \psi)$
$\pi_{mix} \leftarrow \mathsf{Proof}_{mix}((\mathbf{pk}, \mathbf{pk}'), (\mathbf{r}', \psi))$
Output$(\mathbf{pk}', \mathbf{r}', \pi_{mix})$

Reporting

$\mathsf{Report}(id, sk, ek, m, t)$:
$m_{\mathsf{bin}} = (m_1, \ldots, m_b)_2, m_i \in \{0, 1\}$
for $i = 1$ to b **do**
$ct_i \leftarrow \mathsf{Enc}(ek, m_i; r_i)$
$\pi_{m_i} \leftarrow \mathsf{Proof}_m(ct_i, (r_i, m_i))$
end for
$\mathbf{ct} \leftarrow (ct_1, \ldots, ct_b)$
$\boldsymbol{\pi_m} \leftarrow (\pi_{m_1}, \ldots, \pi_{m_b})$
$\sigma \leftarrow \mathsf{Sign}(sk, (t, \mathbf{ct}, \boldsymbol{\pi_m}))$
Output $(pk, (t, \mathbf{ct}, \boldsymbol{\pi_m}), \sigma)$

Report Anonymization

$\mathsf{Anonym}(\{(pk_i, t, \mathbf{ct}_i, \boldsymbol{\pi_{m_i}}, \sigma_i)\}_{i\in M}, \{r'_i\}_{i\in M}, sk_{EA})$
for each $i \in M$ **do**
$x_i = (pk'_i, \mathbf{ct}_i, \boldsymbol{\pi}_{m_i}, t)$
$\omega_i = (pk_i, \sigma_i, r'_i)$
$\pi_i \leftarrow \mathsf{Proof}_{\sigma,r}(x_i, \omega_i)$
end for
$\sigma_{EA} \leftarrow \mathsf{Sign}(sk_{EA}, H((pk'_i, \mathbf{ct}_i, \boldsymbol{\pi}_{m_i}, t, \pi_i)_{i\in M}))$
Output $(\{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi_{m_i}}, t, \pi_i\}_{i\in M}, \sigma_{EA})$

Validation

$\mathsf{Validate}(\mathcal{BB}, \{pk'_i, \mathbf{ct_i}, \boldsymbol{\pi_{m_i}}, t, \pi_i\})_{i\in M}$:
Verify σ_{EA}, freshness, $\mathsf{Verify}_{\sigma,r}$, and Verify_m
Output $\top/\perp$

Evaluation (threshold dk**)**

$\mathsf{Eval}(\mathcal{BB})$:
For each $(\mathbf{ct_m}, \mathbf{ct_b}) \in \mathcal{BB}$:
$(ct_o, \pi_{ord}) \leftarrow \mathsf{ord}(\mathbf{ct_m}, \mathbf{ct_b})$
$(\mathbf{ct_{red}}, \pi_{dec}, 1/g_i) \leftarrow \mathsf{Reduct}(\mathbf{ct_b}, \mathbf{ct_m}, ct_o)$
set $\mathbf{CT_{red}} = \{(\mathbf{ct_{red}}, t, pk')\}_{pk'\in\mathbf{pk}'}$
$ct_{\mathsf{sum}} \leftarrow \mathsf{Agg}(\mathbf{CT_{red}})$
For each $ct_{T_i} \in \mathbf{ct_T}$:
$(ct_{eq_i}, \pi_{r_i}) \leftarrow \mathsf{Epet}(ct_{\mathsf{sum}}, ct_{T_i})$
Set $\boldsymbol{\pi_{eq}} = \{\pi_{r_i}\}_{i\in T}$ and $\mathbf{ct_{eq}} = \{ct_{eq_i}\}_{i\in T}$
Output $(\mathbf{ct_{eq}}, \boldsymbol{\pi}_{eq})$

For each $ct_{eq_i} \in \mathbf{ct}_{eq}$:
$r_i \leftarrow \mathsf{Dec}(dk, ct_{eq_i})$
$\pi_i \xleftarrow{\$} \mathsf{Proof}_{dec}(ek, r_i, ct_{eq_i}), dk)$
Set $\boldsymbol{R} = (r_i)_{i\in T}$, $\boldsymbol{\pi_{dec}} = (\pi_i)_{i\in T}$
Output $(\boldsymbol{R}, \boldsymbol{\pi_{dec}})$

Verification

$\mathsf{Verify}(\mathcal{BB}, \boldsymbol{R}, \Pi)$:
Verify $\Pi = (\pi_{ord}, \boldsymbol{\pi_{dec}}, \boldsymbol{\pi_{eq}}, (\pi_i)_{i\in T})$, all $\pi_{sk}, \pi_{dk}, \boldsymbol{\pi_{m_i}}, \pi_i, \pi_{mix}$, and σ_{EA}
Output $\top/\perp$