Privacy-Preserving Aggregation of Time-Series Data with Public Verifiability from Simple Assumptions*

Keita Emura[§]

[§]National Institute of Information and Communications Technology (NICT), Japan.
k-emura@nict.go.jp

May 29, 2017

Abstract

Aggregator oblivious encryption was proposed by Shi et al. (NDSS 2011), where an aggregator can compute an aggregated sum of data and is unable to learn anything else (aggregator obliviousness). Since the aggregator does not learn individual data that may reveal users' habits and behaviors, several applications, such as privacy-preserving smart metering, have been considered. In this paper, we propose aggregator oblivious encryption schemes with public verifiability where the aggregator is required to generate a proof of an aggregated sum and anyone can verify whether the aggregated sum has been correctly computed by the aggregator. Though Leontiadis et al. (CANS 2015) considered the verifiability, their scheme requires an interactive complexity assumption to provide the unforgeability of the proof. Our schemes are proven to be unforgeable under a static and simple assumption (a variant of the Computational Diffie-Hellman assumption). Moreover, our schemes inherit the tightness of the reduction of the Benhamouda et al. scheme (ACM TISSEC 2016) for proving aggregator obliviousness. This tight reduction allows us to employ elliptic curves of a smaller order and leads to efficient implementation.

1 Introduction

1.1 Aggregator Oblivious Encryption

Aggregator oblivious encryption was proposed by Shi et al. [49], where an aggregated sum of n users' data (such as energy consumption from smart meters) can be computed in a privacy-preserving manner. In brief, an honest dealer generates secret keys for users and an aggregator. A user i encrypts data $x_{i,t}$ at time t, and sends the ciphertext $c_{i,t}$ to the aggregator. The aggregator can compute the aggregated sum $X_t = \sum_{i=1}^{n} x_{i,t}$ from $\{c_{i,t}\}_{i\in[1,n]}$ and sends X_t to a data analyzer (such as an energy provider). It is particularly worth noting that the aggregator learns X_t and nothing else and this security notion has been formalized as aggregator obliviousness. Note that if homomorphic encryption [23, 46] is simply employed, then the aggregator has the capability to decrypt each $c_{i,t}$ and can obtain $x_{i,t}$. Since $x_{i,t}$ may reveal consumer habits and behaviors, e.g., when a certain consumer turns the air conditioner on, it may appear when the consumer returns home, aggregator oblivious encryption is better to preserve the privacy of users. Moreover, the aggregator is not required to be a fully trusted authority and is modeled as honest-but-curious. That is, the data analyzer can collect the aggregated sum of $x_{i,t}$ via the aggregator in a privacy-preserving manner. In addition, only a unidirectional channel is required from each user to the aggregator. This could be an advantage compared to the schemes that require bidirectional channels between the smart meters and the aggregator [47, 22]. Though the Shi et al. scheme is not tolerant of user failures (i.e., if even a single user fails to respond in a certain aggregation round, the aggregation algorithm does not work), Chan et al. [13] proposed a fault-tolerant solution such that the aggregator can still compute the aggregated sum from the remaining users.

*An extended abstract appears in the 22nd Australasian Conference on Information Security and Privacy (ACISP 2017) [19].

The Shi et al. scheme is aggregator obliviousness under the Decisional Diffie-Hellman (DDH) assumption in the random oracle model. They employed the lifted ElGamal encryption approach [14] and therefore $X_t = \sum_{i=1}^{n} x_{i,t}$ needs to be suitably small since the aggregator is required to solve the discrete logarithm g^{X_t} with respect to basis g. Later, Joye and Libert [31] proposed an aggregator oblivious encryption scheme with large plaintext spaces by employing the Paillier-type homomorphic operation [46]. The Joye-Libert scheme is aggregator obliviousness under the Decision Composite Residuosity (DCR) assumption in the random oracle model. Both schemes [31, 49] were generalized by Benhamouda, Joye, and Libert (BJL) [9]. They gave a generic construction of aggregator oblivious encryption from smooth projective hash functions [15] with an extra additively homomorphic property over the key space, with both DDH and DCR-based instantiations. An attractive point of the BJL construction is its tight reduction. Namely, the reduction loss is $O(t_{\mathsf{max}})$ whereas that of the Shi et al. scheme [49] is $O(t_{\mathsf{max}} n^3)$ where t_{max} is the maximum time to be supported by the system and n is the number of users. If we consider the exact security [8, 42], then tight reduction is important. As in Benhamouda et al. [9], we set that $n = t_{\mathsf{max}} = 2^{20} \approx 10^6$ which approximately allows the computation of an aggregation every 15 minutes for 30 years throughout a city like Paris. Then, the security loss of the Shi et al. scheme is approximately 2^{80}. That is, That is, for achieving 112-bit security, the Shi et al. scheme requires approximately 7,680-bit public key or elliptic curves with 384–511-bit order, which is recommended by NIST [5] for achieving 192-bit security. On the other hand, the security loss of the Benhamouda et al. scheme is approximately 2^{20}, and to achieve 112-bit security, approximately 3,072-bit public key or elliptic curves with 256–383-bit order is required.[1]

1.2 Aggregator Oblivious Encryption with Public Verifiability

As mentioned above, the aggregator is modeled as honest-but-curious and is assumed to output X_t correctly. For stronger security, Leontiadis et al. [36] considered a new model: a user i produces a tag $\sigma_{i,t}$ in addition to $c_{i,t}$, and sends $(c_{i,t}, \sigma_{i,t})$ to the aggregator, and the aggregator is required to generate a publicly verifiable proof σ_t that proves the decryption result of $\{c_{i,t}\}_{i\in[1,n]}$ is exactly X_t. Of course, it is required that the aggregator cannot produce a forged σ_t for some $X_t \neq \sum_{i=1}^{n} x_{i,t}$, and this security notion is formalized as aggregator unforgeability. Since the data analyzer can recognize whether the aggregator correctly computed the aggregated sum, this functionality can be seen as a kind of verifiable computation [3, 21].

Though the Leontiadis et al. approach is interesting, one drawback of their construction is the underlying complexity assumption. They introduced an interactive assumption called the LEOM assumption for proving aggregator unforgeability. The LEOM assumption is defined as follows.

Definition 1 (LEOM Assumption [36]) *Let $D = (p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be bilinear groups. Choose $\alpha \xleftarrow{\$} \mathbb{G}_1$ and $\delta, \gamma_1, \ldots, \gamma_n \xleftarrow{\$} \mathbb{Z}_p$ and set $\Gamma = g_2^{\gamma}$ and $\Delta = g_2^{\sum_{i=1}^{n}\gamma_i}$. The LEOM oracle $\mathcal{O}_{\mathsf{LEOM}}$ takes as input $(t, \{x_{i,t}\}_{i=1}^{n})$, chooses $\beta_t \xleftarrow{\$} \mathbb{G}_1$, and returns $(\alpha, \beta_t, \{\beta_t^{\gamma_i}\alpha^{\delta x_{i,t}}\}_{i=1}^{n})$. If a query at t contains $i' \in [1, n]$ such that $x_{i,t} \neq x'_{i,t}$, then $\mathcal{O}_{\mathsf{LEOM}}$ returns $\perp$. Assume that $\mathcal{O}_{\mathsf{LEOM}}$ is called once at each t. We say that the LEOM assumption holds if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{LEOM}(\lambda) := \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{LEOM}}(\cdot,\cdot)}(D, \Gamma, \Delta) \to (t, z, c)]$ is negligible where $\mathcal{A}$ has queried $(t, \{x_{i,t}\}_{i=1}^{n})$ and $z \neq \sum_{i=1}^{n} x_{i,t}$ and $c = \beta_t^{\sum_{i=1}^{n}\gamma_i}\alpha^{z\delta}$ holds.*

However, as explained by Naor [43], it is better to avoid interactive assumptions as much as possible to prevent circular arguments. Making cryptographic primitives secure under weak assumptions is one of the important topics of cryptography. To name a few, verifiable random functions [27, 28], group signatures [38, 39], structure-preserving signatures [2], identity-based encryption [51], attribute-based encryption [45, 50], oblivious transfer [25] and so on, and constructing an aggregator oblivious encryption scheme with public verifiability from static and simple assumptions are still left as open problems.

[1] This key-length is recommended by NIST [5] for achieving 128-bit security. To be precise, the Benhamouda et al. scheme archives 108-bit security under this key length. Thus, a slightly longer key is required to achieve 112-bit security.

Table 1: Comparison of DL-based Aggregator Oblivious Encryption

Scheme	Ciphertext Size ($c_{i,t}$)	Tag Size ($\sigma_{i,t}$)	Secret Key Size	Public Parameter Size ($\mathsf{params} + \mathsf{vk}$)
BJL (DDH) [9]	$\|\mathbb{G}_1\|$	-	$2\|\mathbb{Z}_p\|$	$\|\mathbb{G}_1\|$+ 2 hash
LEOM [36]	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$+$\|\mathbb{G}_2\|$+ 1 hash
Ours 1	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$3\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$	$\|\mathbb{G}_1\| + (1 + t_{\mathsf{max}})\|\mathbb{G}_2\| + \|\mathbb{G}_T\|$+6 hash[1]
Ours 2	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$	$\|\mathbb{G}_1\| + \|\mathbb{G}_2\| + \|\mathbb{G}_T\|$+5 hash

Scheme	$\|p\|$[‡]	Reduction Loss AO/AU[2]	Encryption Algorithm	Aggregator Unforgeability	Complexity Assumptions for proving AO/AU[2]	Bulletin Board
BJL (DDH) [9]	256	$O(t_{\mathsf{max}})$/-	Deterministic	-	DDH/-	-
LEOM [36]	1031	$O(t_{\mathsf{max}}n^3)/O(1)$	Deterministic	Full	DDH/LEOM[3]	-
Ours 1	383	$O(t_{\mathsf{max}})/O(1)$	Deterministic	Weak	DDH/mCDH[4]	-
Ours 2	383–1031	$O(t_{\mathsf{max}})/O(t_{\mathsf{max}}^2)$	Probabilistic	Semi-Adaptive	DDH/DDH&mCDH[4]	Required

[†] $\mathbb{Z}_p$, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, and $|\mathbb{G}_T|$ denote the bit-length of an element of $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively.

[‡] $|p|$ denotes the bit-length of p for 112-bit security. Here, we set $n = t_{\mathsf{max}} = 2^{20}$ [9]. For the BJL scheme, we refer the NIST recommendation [5] since the BJL scheme is pairing-free. For the LEOM scheme and ours, we refer the result by Menezes, Sarkar, and Singh [41] who re-evaluated parameters of pairing-friendly elliptic curves by considering the result by Kim and Barbulescu [34]. Since it is not clear how large p is required for 152-bit security, we denote $|p|$: 383–1031 for the second scheme.

[1] Remark that no user is required to have the large-size verification key.

[2] AO/AU: Aggregator Obliviousness/Aggregator Unforgeability

[3] LEOM: Leontiadis-Elkhiyaoui-Önen-Molva. An interactive complexity assumption.

[4] mCDH: modified Computational Diffie-Hellman. A static complexity assumption.

1.3 Our Contribution

In this paper, we propose two aggregator oblivious encryption schemes with public verifiability from static and simple assumptions (a variant of the Computational Diffie-Hellman (CDH) assumption). See Table 1 for detailed comparisons. For aggregator obliviousness, both schemes are tightly reduced to the BJL scheme. That is, our schemes inherit the tightness of the reduction of the Benhamouda et al. scheme. This tight reduction allows us to employ elliptic curves with a smaller order and leads to efficient implementation. On the other hand, the Leontiadis et al. scheme is reduced to the Shi et al. scheme and has a loose reduction. Remark that Benhamouda et al. [9] also show that a degradation factor of at least $\Omega(n^2)$ is unavoidable in the Shi et al. scheme. They show that any blackbox nonrewinding reduction from the Shi et al. scheme to a noninteractive problem loses a factor of at least n^2. That is, this bound cannot be improved in the Shi et al. scheme, and the Leontiadis et al. scheme also.

The first scheme provides weak aggregator unforgeability, where an adversary can obtain ciphertexts and tags $\{(c_{i,t}, \sigma_{i,t})\}_{i=1}^n$ of $x_{i,t}$ chosen by the encryption oracle. Note that in the smart meter setting, $x_{i,t}$ (such as power consumption) is measured by the meter. Thus, we believe that weak aggregator unforgeability is still meaningful in the actual usage. One drawback to the first scheme, beside weak aggregator unforgeability, is the large-size verification key $\mathsf{vk} = \{\mathsf{vk}_t := g_2^{\sum_{i=1}^n v_{i,t}}\}_{t\in[1,t_{\mathsf{max}}]}$ where t_{max} is the maximum time to be supported by the system. If we employ Barreto-Naehrig (BN) curves [6] with a 383-bit order, then approximately 100MByte-sized verification keys need to be published when $t_{\mathsf{max}} = 2^{20} \approx 10^6$ [9]. Note that no user is required to have the large-size verification key. Moreover, verification keys for past times can be removed. In addition, if we can assume that these keys are updated by the dealer every time over a certain time period (i.e., periodic inspection of meters every one to two years), or if we can set a relatively small t_{max}, then we can significantly reduce the size of the keys to be stored. Remark that, if a user manages all $v_{i,t}$ as its secret key, then the secret key size also depends on t_{max}. To avoid such a large-size secret key, we additionally introduce a hash function H and a time-independent secret key v_i, and we compute $v_{i,t} = H(v_i, t)$. This helps us to reduce the secret key size. For weak aggregator unforgeability, the first scheme provides a tight reduction loss from the advantage of the the mCDH problem.

Though we can reduce the verification key size according to the t_{max} settings, it would be better to support constant-size keys. Our second scheme solves the large-size key problem by choosing $v_{i,t}$ on the fly. That is, in the second scheme a user i chooses $v_{i,t}$ in the encryption phase, whereas in the first scheme all keys are generated by an honest dealer, as in previous works [49, 9, 31, 36]. Though the Enc algorithm becomes probabilistic, this strategy allows us to prove that the scheme provides aggregator unforgeability with semi-adaptive chosen message attack where an adversary can obtain ciphertexts and tags $\{(c_{i,t}, \sigma_{i,t})\}_{i=1}^{n}$ of $x_{i,t}$ chosen by the adversary. Here, semi-adaptive means that the adversary is required to send all $\{x_{i,t}\}_{i=1}^{n}$, and obtains the corresponding $\{(c_{i,t}, \sigma_{i,t})\}_{i=1}^{n}$. Though vk can be removed from the public value, a drawback of the second scheme is that a malicious aggregator could modify vk. Thus, we additionally need to introduce public channels equipped with memory, such as a bulletin board [26] that is publicly readable and that every user can write to, but nobody can delete from. See Section 4 for a more detailed explanation. Another drawback is its reduction loss. Though aggregator obliviousness of the second scheme is tightly reduced to the BJL scheme (this requires $O(t_{\mathsf{max}})$ reduction loss from the advantage of the DDH problem), semi-adaptive aggregator unforgeability of the second scheme requires additional reduction loss. Concretely, $O(t_{\mathsf{max}}^2)$ reduction loss from the advantage of the DDH problem. Thus, we need to employ elliptic curves with a relatively large order.

1.4 Related Work

Aggregator oblivious encryption considers collecting the aggregated sum of users (e.g., the total consumption of customers) in a certain region for each time period. This could be employed for privacy-preserving energy management systems. On the other hand, collecting the aggregated sum of a particular user might be desired for a certain reason. For example, if an energy provider would like to send an invoice to a customer and would like to know the total amount of the consumption of the customer. This could be employed for privacy-preserving supplier billing systems [29, 48]. Some schemes support both billing and energy management functionality [7, 44, 17]. Ohara et al. [44] in particular proposed such a smart metering scheme with verifiability of the integrity of the total amount of consumption or the billing price.

In our setting (as in [9, 31, 36, 49]), the number of users n is selected and fixed during the setup phase. Some papers considered dynamic joins and leaves [37, 13, 30, 35]. Chan et al. [13] proposed a binary interval tree technique that reduces the communication cost for joins and leaves, and Jawurek et al. [30] further improved the communication overhead of the Chan et al. scheme. Although the Chan et al. and Jawurek et al. schemes require public key settings, Li and Cao [37] proposed a more efficient scheme that only requires symmetric key settings. Though these schemes assume an honest dealer that issues keys to the users and the aggregator via a secure channel, Leontiadis et al. [35] proposed a key update mechanism that does not require any trusted dealer. They introduced an additional semi-trusted party called the collector that collects partial key information from users via a secure channel.

Datta and Joye [18] showed that a protocol for computing an aggregate sum proposed by Jung, Li, and Wan [32] is universally breakable, where anyone can recover private data from ciphertexts.

Some schemes employ bilinear groups with composite order $N = pq$ [40, 20]. This could be a bottleneck since we need to assume that N is difficult to be factorized and is selected as sufficiently large. In the meantime, our schemes are constructed over bilinear groups with a prime order.

Benhamouda et al. [9] mentioned that multi-input functional encryption [24] implies aggregator oblivious encryption. Since Badrinarayanan et al. [4] proposed verifiable functional encryption and also considered its multi-input setting, we might be able to construct verifiable aggregator oblivious encryption from verifiable multi-input functional encryption. Though, as in Benhamouda et al., we leave this attempt in this paper due to the efficiency point of view.

1.5 Differences from the Proceedings Version

In the proceedings version [19], we claimed that the second scheme provides full aggregator unforgeability where an adversary is allowed to adaptively choose $x_{i,t}$, and can obtain the corresponding ciphertext and tag $(c_{i,t}, \sigma_{i,t})$. Intuitively, in the security proof, the simulator responds the encryption query $(i, t, x_{i,t})$ for $i \in [1, n-1]$ by preparing a ciphertext and tag of $r_{i,t}$ for some random $r_{i,t} \in \mathbb{Z}_p$ (regardless of $x_{i,t}$), and for the encryption query $(n, t, x_{n,t})$, the simulator prepares a ciphertext and tag of $\sum_{i=1}^{n} x_{i,t} - \sum_{i=1}^{n-1} r_{i,t}$. Though the decryption result of these ciphertexts is exactly $\sum_{i=1}^{n} x_{i,t}$ that the adversary queried, we need to show that ciphertexts and tags of $r_{i,t}$ (resp. $\sum_{i=1}^{n} x_{i,t} - \sum_{i=1}^{n-1} r_{i,t}$) and those of $x_{i,t}$ (resp $x_{n,t}$)

are indistinguishable. Though we can reduce this indistinguishability to aggregator obliviousness, for simulation, the adversary is required to send all $\{x_{i,t}\}_{i=1}^{n}$. Thus, we re-claim that the second scheme is aggregator unforgeability secure against semi-adaptive chosen message attack. Due to the additional reduction, the reduction loss of the second scheme becomes $O(t_{\mathsf{max}}^2)$. Thus, we need to reconsider the order of the underlying bilinear groups.

2 Preliminaries

Let p is a λ-bit prime, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We use the (type 3) asymmetric setting, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, and no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known.

Next, we define the Decisional Diffie-Hellman (DDH) assumption on $\mathbb{G}_1$ as follows.

Definition 2 (DDH Assumption) *Let $D := (p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, $g'_1 \xleftarrow{\$} \mathbb{G}_1$ and $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$ where $r_1 \neq r_2$. We say that the DDH assumption holds on $\mathbb{G}_1$ if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{DDH}(\lambda) := |\Pr[\mathcal{A}(D, g'_1, g_1^{r_1}, {g'_1}^{r_1}) \to true] - \Pr[\mathcal{A}(D, g'_1, g_1^{r_1}, {g'_1}^{r_2}) \to true]|$ is negligible.*

Next, we define a new complexity assumption. This is a variant of the Computational Diffie-Hellman (CDH) assumption. We call this assumption the modified CDH (mCDH) assumption.[2]

Definition 3 (Modified CDH Assumption) *Let $D := (p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, and $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. We say that the Modified CDH assumption holds if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{mCDH}(\lambda) := \Pr[\mathcal{A}(D, g_1^a, g_1^{1/a}, g_1^b, g_2^a) \to g_1^{ab}]$ is negligible.*

We can check that the mCDH assumption holds in the generic bilinear group model by reducing the mCDH problem to the following problem: given $(g_1, g_1^a, g_1^{a^2}, g_1^b, g_2, g_2^a) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$ for random $a, b \in \mathbb{Z}_p$, compute $e(g_1, g_2)^{a^2 b}$. We can assume that the problem is difficult to be solved since it belongs to the Uber assumption family [11]. This reduction can be easily done by setting $g'_1 := g_1^{1/a}$ and $B := ab$. Then, an instance of the mCDH problem $(g_1, g_1^a, g_1^{1/a}, g_1^b, g_2, g_2^a)$ is represented as: given $({g'_1}^a, {g'_1}^{a^2}, g'_1, {g'_1}^B, g_2, g_2^a)$, compute $g_1^{ab} = {g'_1}^{a^2 b} = {g'_1}^{aB}$. We rewrite it: given $(g_1, g_1^a, g_1^{a^2}, g_1^b, g_2, g_2^a)$, compute g_1^{ab}. That is, if the mCDH problem can be solved, then we can compute $e(g_1^{ab}, g_2^a) = e(g_1, g_2)^{a^2 b}$.

3 Definitions of Verifiable Aggregator Oblivious Encryption

In this section, we give the syntax of verifiable aggregator oblivious encryption and its security definitions (aggregator obliviousness and aggregator unforgeability), and introduce the DDH-based BJL scheme [9]. As in Shi et al. we consider encrypt-once security where each user only encrypts once at each time t.

3.1 Syntax of Verifiable Aggregator Oblivious Encryption

Definition 4 (Verifiable Aggregator Oblivious Encryption [36])

Setup: *The setup algorithm takes as input a security parameter λ, and outputs a public parameter* param *and a secret key of aggregator* sk_A*, a set of user secret keys* $\{\mathsf{sk}_i\}_{i=1}^{n}$*, and the aggregate verification key* vk*. We assume that the maximum time* t_{max} *is contained in* param*, and* t_{max} *is a polynomial of the security parameter. We assume that* $t \in [1, t_{\mathsf{max}}]$ *and the verification key at* t vk_t *is contained in* vk*.*

[2] Kiltz and Vahlis [33] defined the modified Decisional Bilinear Diffie-Hellman (mDBDH) assumption where given $(g, g^x, g^y, g^{y^2}, g^z, Z)$ decide whether $Z = e(g, g)^{xyz}$ or not. That is, compared to the original DBDH assumption, the element g^{y^2} is additionally given to the adversary. In our assumption, if we set $g_1^{1/a} := g'_1$ then $(g_1^{1/a}, g_1, g_1^a)$ can be seen as $(g'_1, {g'_1}^a, {g'_1}^{a^2})$. That is, the element ${g'_1}^{a^2}$ is added to an instance of the CDH assumption. Hence, we call the assumption mCDH.

Enc: *The encryption algorithm takes as input* param*,* t*, a value* $x_{i,t} \in \mathbb{Z}_M$*, and* sk_i*, and outputs a ciphertext* $c_{i,t}$ *and a tag* $\sigma_{i,t}$*. Here,* M *is some fixed integer contained in* param*.*

AggrDec: *The aggregation and decryption algorithm takes as input* param*,* t*, and a set of ciphertexts and tags* $\{(c_{i,t}, \sigma_{i,t})\}_{i=1}^n$*, and* sk_A*, and outputs* $X_t := \sum_{i=1}^n x_{i,t} \bmod M$*, and the proof* σ_t*.*

VerifySum: *The verification of aggregation algorithm takes as input* param*,* t*,* vk_t*, and* (X_t, σ_t)*, and outputs 1 or 0.*

We require the following correctness. For all $(\mathsf{param}, \mathsf{sk}_A, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$, and $(X_t, \sigma_t) \leftarrow \mathsf{AggrDec}(\mathsf{param}, t, \{(c_{i,t}, \sigma_{i,t})\}_{i=1}^n, \mathsf{sk}_A)$, $\mathsf{VerifySum}(\mathsf{param}, t, X_t, \sigma_t, \mathsf{vk}_t) = 1$, and $X_t = \sum_{i=1}^n x_{i,t} \bmod M$ hold.

Let us introduce the entities of the system and how to run the algorithms above as follows. We consider four entities, a trusted dealer, an aggregator, users, and a data analyzer. First, the dealer runs $(\mathsf{param}, \mathsf{sk}_A, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and issues sk_A to the aggregator and sk_i to the user i, respectively, and publishes $(\mathsf{param}, \mathsf{vk})$.[3] At time t, each user i encrypts $x_{i,t}$ such that $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$, and sends $(c_{i,t}, \sigma_{i,t})$ to the aggregator. The aggregator runs $(X_t, \sigma_t) \leftarrow \mathsf{AggrDec}(\mathsf{param}, t, \{(c_{i,t}, \sigma_{i,t})\}_{i=1}^n, \mathsf{sk}_A)$, and sends (X_t, σ_t) to the data analyzer. The data analyzer checks whether the computed aggregated sum X_t is correct by running $1/0 \leftarrow \mathsf{VerifySum}(\mathsf{param}, t, X_t, \sigma_t, \mathsf{vk}_t)$.

3.2 Security Definitions

Next, we define aggregator obliviousness. This requires that the aggregator cannot learn anything more than the aggregate value X_t for each time t. We additionally require that tags $\sigma_{i,t}$ do not affect the security. Let st be state information that $\mathcal{A}$ can preserve any information, and st is used for transferring state information to the other stage. Let $\mathbb{U}$ be the whole set of users for which, at the end of the game, no encryption queries have been made on t^* and no corruption queries have been made. The adversary indicates $\mathbb{S}_{t^*} \subseteq \mathbb{U}$ and obtains $(c_{i,t^*}, \sigma_{i,t^*})$ for all $i \in \mathbb{S}_{t^*}$. Remark that the AggrDec algorithm works only when all ciphertexts are collected. That is, if $\mathbb{S}_{t^*}$ is a proper subset of $\mathbb{U}$ ($\mathbb{S}_{t^*} \subsetneq \mathbb{U}$), then there exist at least one ciphertext c_{i,t^*} such that $i \in \mathbb{U} \setminus \mathbb{S}_{t^*}$. In this case, the adversary cannot run the AggrDec algorithm. Thus, as in the definition of Benhamouda et al. [9] and Shi et al. [49], we require that $\sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(0)} \bmod M = \sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(1)} \bmod M$ must be hold if sk_A is compromised by the adversary and $\mathbb{S}_{t^*} = \mathbb{U}$. Though in the definition of Leontiadis et al. [36], sk_A is always given to the adversary and always the condition $\sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(0)} \bmod M = \sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(1)} \bmod M$ is required, we follow the definition given in [9, 49] where the adversary is allowed to select whether the adversary compromises sk_A or not.

Definition 5 (Aggregator Obliviousness [9, 36]) *For any PPT adversary* $\mathcal{A}$ *and a security parameter* $\lambda \in \mathbb{N}$*, we define the experiment* $\mathrm{Exp}_{\mathcal{A}}^{AO}(\lambda)$ *as follows. If* sk_A *is compromised at the end of the game and* $\mathbb{S}_{t^*} = \mathbb{U}$*, then it is required that* $\sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(0)} \bmod M = \sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(1)} \bmod M$*.*

$$
\begin{aligned}
&\mathrm{Exp}_{\mathcal{A}}^{AO}(\lambda):\\
&\quad (\mathsf{param}, \mathsf{sk}_A, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)\\
&\quad (\mathbb{S}_{t^*}, t^*, \{(x_{i,t^*}^{(0)}, x_{i,t^*}^{(1)})\}_{i \in \mathbb{S}_{t^*}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}, \mathcal{O}_{\mathsf{corrupt}}}(\mathsf{param}, \mathsf{vk}, st);\ \mathbb{S}_{t^*} \subseteq \mathbb{U};\ b \xleftarrow{\$} \{0,1\}\\
&\quad \textit{For all } i \in \mathbb{S}_{t^*}\\
&\quad\quad (c_{i,t^*}, \sigma_{i,t^*}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t^*}^{(b)}, \mathsf{sk}_i)\\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}, \mathcal{O}_{\mathsf{corrupt}}}(\{(c_{i,t^*}, \sigma_{i,t^*})\}_{i \in \mathbb{S}_{t^*}}, st)\\
&\quad \textit{If } b = b',\ \textit{then return } 1 \textit{ and } 0 \textit{ otherwise}
\end{aligned}
$$

- $\mathcal{O}_{\mathsf{enc}}$*: This encryption oracle takes as input a tuple* $(i, t, x_{i,t})$*, and returns* $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$*. Note that* $\mathcal{A}$ *is not allowed to input* $(i, t^*, \cdot)$ *where* $i \in \mathbb{S}_{t^*}$ *to this oracle.*

[3] In the definition of Leontiadis et al. [36], each user i chooses a tag value tk_i, and sends its encoding value to the dealer in the Setup phase. The dealer computes vk from all tk_i. Here we simply assume that vk is generated by the dealer since the dealer is modeled as a trusted entity. Later, we consider the case that vk is generated by users in the encryption phase.

- $\mathcal{O}_{\mathsf{corrupt}}$: *This corruption oracle takes as input* $i \in [0, n]$, *and returns* sk_i. *If* $i = 0$, *then the oracle returns* sk_A. *Note that* $\mathcal{A}$ *is not allowed to input* $i \in \mathbb{S}_{t^*}$ *to this oracle.*

We say that an encryption scheme is aggregator obliviousness if the advantage $\mathrm{Adv}_{\mathcal{A}}^{AO}(\lambda) := 2|\Pr[\mathrm{Exp}_{\mathcal{A}}^{AO}(\lambda) = 1] - 1/2|$ *is negligible for any PPT adversary* $\mathcal{A}$.

Next, we define aggregator unforgeability. This requires that an adversary (modeled as the malicious aggregator) cannot produce a forged tag σ_t that is accepted by the $\mathsf{VerifySum}$ algorithm. As in the definition of unforgeability given by Leontiadis et al. [36], we consider two cases: an adversary is required either the adversary does not obtain ciphertexts and tags at the challenge time t^* (type I forgery) or the adversary has obtained all ciphertexts and tags $\{(c_{i,t^*}, \sigma_{i,t^*})\}_{i=1}^n$ (type II forgery). In the type II forgery case, it is assumed that ciphertexts and tags are honestly generated, and $\mathcal{A}$ obtains ciphertexts and tags of all users in the system. Type I adversary captures the case that the aggregator tries to generate a forged tag σ_t at a future time t (i.e., users have not generated $(c_{i,t}, \sigma_{i,t})$). Type II adversary captures the case that the aggregator tries to generate a forged tag σ_t at a past/current time t (i.e., users have generated $(c_{i,t}, \sigma_{i,t})$).

Definition 6 (Aggregator Unforgeability [36]) *For any PPT adversary* $\mathcal{A}$ *and a security parameter* $\lambda \in \mathbb{N}$, *we define the experiment* $\mathrm{Exp}_{\mathcal{A}}^{AU}(\lambda)$ *as follows.*

$$\begin{aligned}
&\mathrm{Exp}_{\mathcal{A}}^{AU}(\lambda): \\
&\quad (\mathsf{param}, \mathsf{sk}_A, \{\mathsf{sk}_i\}_{i=1}^n, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda) \\
&\quad (t^*, X_{t^*}, \sigma_{t^*}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}}(\mathsf{param}, \mathsf{sk}_A, \mathsf{vk}) \\
&\quad \textit{If one of the followings hold, then return } 1 \textit{ and } 0 \textit{ otherwise} \\
&\quad \textit{(Type I)}: \ \mathsf{VerifySum}(\mathsf{param}, t^*, X_{t^*}, \sigma_{t^*}, \mathsf{vk}_{t^*}) = 1 \\
&\quad \qquad \qquad \wedge \textit{ No encryption oracle is called at } t^* \\
&\quad \textit{(Type II)}: \ \mathsf{VerifySum}(\mathsf{param}, t^*, X_{t^*}, \sigma_{t^*}, \mathsf{vk}_{t^*}) = 1 \\
&\quad \qquad \qquad \wedge X_{t^*} \neq \sum_{i=1}^{n} x_{i,t^*} \bmod M
\end{aligned}$$

- $\mathcal{O}_{\mathsf{enc}}$: *This encryption oracle takes as input a tuple* $(i, t, x_{i,t})$, *and returns* $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$.

We say that an encryption scheme is aggregator unforgeable if the advantage $\mathrm{Adv}_{\mathcal{A}}^{AU}(\lambda) := \Pr[\mathrm{Exp}_{\mathcal{A}}^{AU}(\lambda) = 1]$ *is negligible for any PPT adversary* $\mathcal{A}$.

Next, we slightly weaken the definition of Leontiadis et al. in the following. In their definition, the adversary (modeled as the malicious aggregator) can adaptively choose $x_{i,t}$ and can obtain the corresponding $(c_{i,t}, \sigma_{i,t})$ from the encryption oracle. This definition is an analogy of Existential Unforgeability against Chosen Message Attack (EUF-CMA) in the signature context where an adversary is allowed to obtain signatures on messages which are (adaptively) chosen by the adversary. However, in the actual situation, the aggregator does not decide $x_{i,t}$, and just receives $c_{i,t}$ sent from users. Actually, in the smart meter setting, $x_{i,t}$ (such as power consumption) is measured by the meter. Thus, it seems reasonable to propose that the adversary just queries (i, t) to the encryption oracle, and the oracle chooses $x_{i,t}$ and returns the corresponding $(c_{i,t}, \sigma_{i,t})$ to the adversary. Our definition is an analogy of Existential Unforgeability against Random Message Attack (EUF-RMA) in the signature context where an adversary is given signatures on randomly chosen messages.

Definition 7 (Weak Aggregator Unforgeability) *For any PPT adversary* $\mathcal{A}$ *and a security parameter* $\lambda \in \mathbb{N}$, *the experiment* $\mathrm{Exp}_{\mathcal{A}}^{wAU}(\lambda)$ *is the same as* $\mathrm{Exp}_{\mathcal{A}}^{AU}(\lambda)$ *except* $\mathcal{O}_{\mathsf{enc}}$.

- $\mathcal{O}_{\mathsf{enc}}$: *This encryption oracle takes as input a tuple* (i, t). *The oracle chooses* $x_{i,t}$ *and returns* $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$.

We say that an encryption scheme is weakly aggregator unforgeable if the advantage $\mathrm{Adv}_{\mathcal{A}}^{wAU}(\lambda) := \Pr[\mathrm{Exp}_{\mathcal{A}}^{wAU}(\lambda) = 1]$ *is negligible for any PPT adversary* $\mathcal{A}$.

Next, we introduce semi-adaptive aggregator unforgeability which is stronger than the weak one but is weaker than the full aggregator unforgeability.

Definition 8 (Semi-Adaptive Aggregator Unforgeability) *For any PPT adversary* $\mathcal{A}$ *and a security parameter* $\lambda \in \mathbb{N}$, *the experiment* $\mathrm{Exp}_{\mathcal{A}}^{saAU}(\lambda)$ *is the same as* $\mathrm{Exp}_{\mathcal{A}}^{AU}(\lambda)$ *except* $\mathcal{O}_{\mathsf{enc}}$.

- $\mathcal{O}_{\mathsf{enc}}$: *This encryption oracle takes as input tuples* $\{(i, t, x_{i,t})\}_{i=1}^{n}$. *The oracle computes* $(c_{i,t}, \sigma_{i,t}) \leftarrow \mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$ *for all* $i \in [1, n]$, *and returns* $\{(c_{i,t}, \sigma_{i,t})\}_{i=1}^{n}$.

We say that an encryption scheme is semi-adaptive aggregator unforgeable if the advantage $\mathrm{Adv}_{\mathcal{A}}^{saAU}(\lambda) := \Pr[\mathrm{Exp}_{\mathcal{A}}^{saAU}(\lambda) = 1]$ *is negligible for any PPT adversary* $\mathcal{A}$.

3.3 The DDH-based BJL Scheme

Benhamouda, Joye, and Libert (BJL) [9] gave a generic construction of aggregator oblivious encryption from smooth projective hash functions [15]. Here, we introduce its DDH instantiation. The underlying idea is essentially the same as that of the She et al. aggregator oblivious encryption. The aggregator has keys (s_0, t_0) where $s_0 + \sum_{i=1}^{n} s_i = 0$ and $t_0 + \sum_{i=1}^{n} t_i = 0$, and this structure allows the aggregator to cancel out a part of ciphertext $H_1(t)^{\sum_{i=1}^{n} s_i}$ and $H_2(t)^{\sum_{i=1}^{n} t_i}$.

Setup: Let $\mathbb{G}_1$ be a DDH-hard group with λ-bit prime order $p = M$ and g_1 be a generator of $\mathbb{G}_1$.

Let $H_i : \mathbb{Z} \to \mathbb{G}_1$ $(i = 1, 2)$ be hash functions. Choose $s_1, \ldots, s_n, t_1, \ldots, t_n \xleftarrow{\$} \mathbb{Z}_p$, set $s_0 = -\sum_{i=1}^{n} s_i$ and $t_0 = -\sum_{i=1}^{n} t_i$. Output $\mathsf{param} = ((p, g_1, \mathbb{G}_1), H_1, H_2)$, $\mathsf{sk}_A = (s_0, t_0)$ and $\mathsf{sk}_i = (s_i, t_i)$.

Enc: Parse $\mathsf{sk}_i = (s_i, t_i)$. For $x_{i,t} \in \mathbb{Z}_p$, compute $c_{i,t} = g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i}$ and output $c_{i,t}$.

$\mathsf{AggrDec}$: Parse $\mathsf{sk}_A = (s_0, t_0)$. Compute $V_t = H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^{n} c_{i,t} = g_1^{X_t}$ where $X_t = \sum_{i=1}^{n} x_{i,t}$, and solve the discrete logarithm V_t with respect to basis g_1. Output X_t.

4 Proposed Constructions

In this section, we propose two schemes. For aggregator obliviousness, both schemes are tightly reduced to the DDH-based BJL scheme. The first scheme only provides weak aggregator unforgeability, whereas the second scheme provides semi-adaptive aggregator unforgeability. The unforgeability of both schemes relies on the mCDH assumption and the second scheme additionally requires public channels with memory, such as a bulletin board [26] (which is publicly readable, and every user can write to, but nobody can delete from). Moreover, users are required to generate random numbers in the Enc algorithm. Thus, the Enc algorithm in the second scheme is probabilistic whereas that of the first scheme is deterministic.

4.1 High-level Description

Aggregator Obliviousness: We employ (type 3) elliptic curves where $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known. Then, we run the BJL scheme [9] over the DDH-hard group $\mathbb{G}_1$, and borrow the ciphertext form $c_{i,t}$ and secret keys sk_A and sk_i. Since the BJL scheme is aggregator obliviousness under the DDH assumption, we can expect that our scheme is also aggregator obliviousness. In order to directly reduce the aggregator obliviousness of our scheme to that of the BJL scheme, we independently prepare the verification part. That is, we introduce $v_{i,t}$ for each user i and in the security proof, $v_{i,t}$ can be chosen independently from the BJL scheme. This setting allows us to compute the tag $\sigma_{i,t}$ from $c_{i,t}$ and $v_{i,t}$ in the security proof. More precisely, the challenge ciphertexts and tags of our scheme $\{(c_{i,t^*}, \sigma_{i,t^*})\}_{i \in \mathbb{S}_{t^*}}$ can be constructed from the challenge ciphertext of the BJL scheme $\{c_{i,t^*}\}_{i \in \mathbb{S}_{t^*}}$ and the corresponding $v_{i,t}$. Thus, we can construct an algorithm that breaks the aggregator obliviousness of the BJL scheme by using an adversary of our scheme. Remark that $\sigma_{i,t}$ has the similar form of $c_{i,t}$ in our scheme due to this reason. This strategy has been considered by Leontiadis et al. [36]. They provided a reduction of their scheme to the Shi et al. scheme [49]. However, as mentioned by Benhamouda et al. [9], the security loss is $O(t_{\mathsf{max}} n^3)$ in the Shi et al. scheme, whereas it is $O(t_{\mathsf{max}})$ in the BJL scheme. Thus, we have chosen the BJL scheme as the underlying scheme in this paper.

Aggregator Unforgeability: For public verification, we pay attention to that the form of the ciphertext $c_{i,t}$ of the BJL scheme is similar to a decryption key of the Boneh-Boyen identity-based encryption (IBE) scheme [10].[4] Due to the above reason, the tag $\sigma_{i,t}$ has the similar form of $c_{i,t}$ in our schemes. Since secure IBE implies a signature [16] (informally, ID is regarded as a message to be signed, and its decryption key is regarded as a signature), we can expect that $\sigma_{i,t}$ is unforgeable. However, to utilize the Boneh-Boyen technique, X_t needs to be embedded into vk in the security proof. Here, we have two choices: whether vk is fixed in the setup phase or not. If vk is chosen by the honest dealer and is fixed in the setup phase, X_t is also required to be fixed in the setup phase (to utilize the security proof technique of selective-ID security of Boney-Boyen IBE), and therefore only weak aggregator unforgeability is provided. Moreover, since one X_t is embedded with one vk, long verification keys is also required where the size lineally depends on t_{max}. We set $\mathsf{vk} = \{\mathsf{vk}_t\}_{t\in[1,t_{\mathsf{max}}]}$ and $\mathsf{vk}_t := g_2^{\sum_{i=1}^n v_{i,t}}$ for $t \in [1, t_{\mathsf{max}}]$. We remark that no user is required to have the large-size verification key. Moreover, if a user i manages all $v_{i,t}$ for $t \in [1, t_{\mathsf{max}}]$ as its secret key sk_i, the secret key size also depends on t_{max}. To avoid such a large-size secret key, we additionally introduce a hash function H and a time-independent secret key v_i, and we compute $v_{i,t} = H(v_i, t)$. That is, in the scheme $v_{i,t}$ is computed by $H(v_i, t)$ whereas in the security proof, $v_{i,t}$ is selected so as to utilize the Boneh-Boyen technique, and set $H(v_i, t) := v_{i,t}$. This helps us to reduce the secret key size.

4.2 The Proposed Scheme 1: Providing Weak Aggregator Unforgeability

We give the first scheme as follows. As mentioned above, vk is chosen in the setup phase.

$\mathsf{Setup}(1^\lambda)$**:** Choose $(p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of λ-bit prime order $p = M$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. Let $H : \mathbb{Z}_p \times [1, t_{\mathsf{max}}] \to \mathbb{Z}_p$ and $H_i : \mathbb{Z} \to \mathbb{G}_1$ $(i = 1, 2, 3, 4, 5)$ be hash functions. Choose $\gamma, s_1, \ldots, s_n, t_1, \ldots, t_n, v_1, \ldots, v_n \xleftarrow{\$} \mathbb{Z}_p$, compute $v_{i,t} = H(v_i, t)$ for all $i \in [1, n]$ and $t \in [1, t_{\mathsf{max}}]$ and set $s_0 = -\sum_{i=1}^n s_i$, $t_0 = -\sum_{i=1}^n t_i$, $h = g_1^\gamma$, and $Z = e(h, g_2)$. Output $\mathsf{param} = ((p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), Z, H, H_1, H_2, H_3, H_4, H_5)$, $\mathsf{sk}_A = (s_0, t_0)$, $\mathsf{sk}_i = (s_i, t_i, v_i, h)$, and $\mathsf{vk} = \{\mathsf{vk}_t\}_{t\in[1,t_{\mathsf{max}}]}$ where $\mathsf{vk}_t = g_2^{\sum_{i=1}^n v_{i,t}}$.

$\mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i)$**:** Parse $\mathsf{sk}_i = (s_i, t_i, v_i, h)$. Compute

$$v_{i,t} = H(v_i, t),\ c_{i,t} = g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i},\ \text{and } \sigma_{i,t} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}}$$

and output $(c_{i,t}, \sigma_{i,t})$.

$\mathsf{AggrDec}(\mathsf{param}, t, \{(c_{i,t}, \sigma_{i,t})\}_{i=1}^n, \mathsf{sk}_A)$**:** Parse $\mathsf{sk}_A = (s_0, t_0)$. Compute

$$V_t = H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^n c_{i,t} = g_1^{X_t}$$

where $X_t = \sum_{i=1}^n x_{i,t}$, and solve the discrete logarithm V_t with respect to basis g_1. Moreover, compute

$$\sigma_t = H_3(t)^{s_0} H_4(t)^{t_0} \prod_{i=1}^n \sigma_{i,t}$$

Output (X_t, σ_t).

[4] A decryption key of the Boneh-Boyen IBE scheme is informally described as $(g^\alpha H_{\mathsf{BB}}(ID)^r, g^r)$ for a master key α and a random r, the Boneh-Boyen hash H_{BB}. In our first construction, α, ID, and r are regarded as $x_{i,t}$, t, and $v_{i,t}$ respectively. Thus, the number of verification keys depends on t_{max}.

$\mathsf{VerifySum}(\mathsf{param}, t, X_t, \sigma_t, \mathsf{vk}_t)$**:** Output 1 if

$$\frac{e(\sigma_t, g_2)}{e(H_5(t), \mathsf{vk}_t)} = Z^{X_t}$$

holds. Otherwise, output 0.

The correctness cleary holds from the following equations.

$$\begin{aligned}
H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^{n} c_{i,t} &= H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^{n} g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \\
&= H_1(t)^{s_0 - \sum_{i=1}^{n} s_i} H_2(t)^{t_0 - \sum_{i=1}^{n} t_i} g_1^{\sum_{i=1}^{n} x_{i,t}} \\
&= g_1^{X_t} \\
\sigma_t &= H_3(t)^{s_0} H_4(t)^{t_0} \prod_{i=1}^{n} \sigma_{i,t} \\
&= H_3(t)^{s_0} H_4(t)^{t_0} \prod_{i=1}^{n} h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}} \\
&= h^{X_t} H_5(t)^{\sum_{i=1}^{n} v_{i,t}} \\
e(\sigma_t, g_2) &= e(h^{X_t} H_5(t)^{\sum_{i=1}^{n} v_{i,t}}, g_2) = e(h, g_2)^{X_t} e(H_5(t), g_2^{\sum_{i=1}^{n} v_{i,t}}) \\
&= Z^{X_t} e(H_5(t), \mathsf{vk}_t)
\end{aligned}$$

Theorem 4.1 *Our scheme 1 is aggregator obliviousness under the DDH assumption on $\mathbb{G}_1$ in the random oracle model.*

We consider the following two games. Game 0 is the original game. Game 1 is the same as Game 0 except that H_3 and H_4 are computed as $H_3(t) = H_1(t)^\gamma$ and $H_4(t) = H_2(t)^\gamma$ for some $\gamma \in \mathbb{Z}_p$. Since $(H_1(t), H_2(t), H_3(t), H_4(t))$ is a DDH tuple, this modification does not affect the security under the DDH assumption on $\mathbb{G}_1$. Briefly, let $(g_1, g_1', g_1^{r_1}, {g_1'}^{r_2}) \in \mathbb{G}_1^4$ be an DDH instance on $\mathbb{G}_1$. For $t \in [1, t_{\mathsf{max}}]$, choose $\tilde{t}_1, \tilde{t}_2 \xleftarrow{\$} \mathbb{Z}_p$, and set $H_1(t) := g_1^{\tilde{t}_1}$, $H_2(t) := {g_1'}^{\tilde{t}_2}$, $H_3(t) := (g_1^{r_1})^{\tilde{t}_1}$, and $H_4(t) := ({g_1'}^{r_2})^{\tilde{t}_2}$. Clearly, if the instance is not a DDH tuple, i.e., $r_1 \neq r_2$, then we simulate Game 0, and if the instance is a DDH tuple, i.e., $r_1 = r_2$, then we simulate Game 1. In Game 1, we construct an algorithm $\mathcal{B}$ that breaks aggregator obliviousness of the BJL scheme as follows.

Proof: Let $\mathcal{A}$ be the adversary of our scheme, and $\mathcal{C}$ be the challenger of the BJL scheme. We construct an algorithm $\mathcal{B}$ that breaks aggregator obliviousness of the BJL scheme as follows. First, $\mathcal{C}$ prepares $(p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, H_1, H_2)$ and sends it to $\mathcal{B}$. $\mathcal{B}$ chooses $\gamma, v_1, \dots, v_n, v_{1,1}, \dots, v_{n,t_{\mathsf{max}}} \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{B}$ computes $h = g_1^\gamma$, $Z = e(h, g_2)$, $\mathsf{vk}_{i,t} = g_2^{v_{i,t}}$ for $i \in [1, n]$ and $t \in [1, t_{\mathsf{max}}]$, and $\mathsf{vk}_t = g_2^{\sum_{i=1}^{n} v_{i,t}}$. $\mathcal{B}$ sets $H(v_i, t) := v_{i,t}$ for $i \in [1, n]$ and $t \in [1, t_{\mathsf{max}}]$. Remark that if $\mathcal{A}$ sends a hash query t, then $\mathcal{B}$ forwards it to $\mathcal{C}$ when $\mathcal{A}$ requests $H_1(t)$ or $H_2(t)$. For H_3 and H_4, $\mathcal{B}$ sets $H_3(t) = H_1(t)^\gamma$ and $H_4(t) = H_2(t)^\gamma$, and returns the hash values. For H_5, $\mathcal{B}$ just returns a random value. $\mathcal{B}$ sends $\mathsf{param} = ((p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), Z, H, H_1, H_2, H_3, H_4, H_5)$, $\{\mathsf{vk}_t\}_{t \in [1, t_{\mathsf{max}}]}$, and vk to $\mathcal{A}$.

If $\mathcal{A}$ sends an encryption query $(i, t, x_{i,t})$ to $\mathcal{B}$, then $\mathcal{B}$ forwards it to $\mathcal{C}$ as an encryption oracle, and obtains $c_{i,t}$. $\mathcal{B}$ computes $c_{i,t}^\gamma H_5(t)^{v_{i,t}} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}}$, and returns $(c_{i,t}, \sigma_{i,t})$ to $\mathcal{A}$. If $\mathcal{A}$ sends a corruption query $i \in [0, n]$ to $\mathcal{B}$, $\mathcal{B}$ forwards it to $\mathcal{C}$ as a corruption query, and obtains sk_A (if $i = 0$) or (s_i, t_i) (if $i \in [1, n]$). If $i = 0$, then $\mathcal{B}$ returns sk_A to $\mathcal{A}$. If $i \in [1, n]$, then $\mathcal{B}$ sets $\mathsf{sk}_i = (s_i, t_i, v_i, h)$, and returns sk_i to $\mathcal{A}$. We remark that if $\mathcal{A}$ sends a hash query (v_i, t), then $\mathcal{B}$ responds $v_{i,t}$ to $\mathcal{A}$.

In the challenge phase, $\mathcal{A}$ sends $(\mathbb{S}_{t^*}, t^*, \{(x_{i,t^*}^{(0)}, x_{i,t^*}^{(1)})\}_{i \in \mathbb{S}_{t^*}})$ to $\mathcal{B}$. Then, $\mathcal{B}$ forwards it to $\mathcal{C}$ as the challenge, and obtains $\{c_{i,t^*}\}_{i \in \mathbb{S}_{t^*}}$. As in the response of encryption queries, $\mathcal{B}$ computes $\sigma_{i,t^*} = c_{i,t^*}^\gamma H_5(t)^{v_{i,t^*}}$ for $i \in \mathbb{S}_{t^*}$, and returns $\{(c_{i,t^*}, \sigma_{i,t^*})\}_{i \in \mathbb{S}_{t^*}}$ to $\mathcal{A}$.

$\mathcal{B}$ responds queries sent from $\mathcal{A}$ as in the previous phase. Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b' and then $\mathcal{B}$ can break aggregator obliviousness of the BJL scheme with the same advantage of $\mathcal{A}$. This concludes the proof since the BJL scheme is aggregator obliviousness under the DDH assumption on $\mathbb{G}_1$ in the random oracle model. □

Theorem 4.2 *Our scheme 1 is weakly aggregator unforgeable under the mCDH assumption in the random oracle model.*

For the proof of Type I forgery, we employ the following assumption: given (g_1^a, g_1^b, g_2^a) compute g_1^{ab}. Since this is equivalent to the CDH assumption if the symmetric pairing setting is employed, we simply call the assumption the CDH assumption in this paper. Remark that this is weaker than mCDH since $g_1^{1/a}$ is not contained in the instance. Since no encryption oracle is called at t^*, the proof is relatively easy. We embed the instance g_1^a to $v_{i,t}$ and g_1^b to the response of the random oracle H_5 respectively. At time t^*, $\mathcal{A}$ outputs (σ_{t^*}, X_{t^*}). From the verification equation, (σ_{t^*}, X_{t^*}) must satisfy $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^n v_{i,t}} h^{X_{t^*}}$. Since $H_5(t^*)^{\sum_{i=1}^n v_{i,t}}$ contains g_1^{ab}, we can solve the CDH problem. Remark that this proof strategy requires $O(t_{\mathsf{max}})$ reduction loss from the advantage of the CDH problem. However, we can achieve a tight reduction (i.e., $O(1)$ reduction loss) from the advantage of the mCDH problem (see below).

For the proof of Type II forgery, our proof strategy is explained as follows. Again, (σ_{t^*}, X_{t^*}) must satisfy $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^n v_{i,t}} h^{X_{t^*}}$. Though $Z = e(h, g_2)$ is published, h itself is not published (contained in sk_i). Thus, we set $h = g_1^{ab}$ and simulate the encryption oracle by using the Boneh-Boyen technique. We embed $1/a$ to $x_{i,t}$ such that $x_{i,t} := x'_{i,t}/a$ for $x'_{i,t} \in \mathbb{Z}_p$. This setting helps us to compute $h^{x_{i,t}} = (g_1^{ab})^{x'_{i,t}/a} = (g_1^b)^{x'_{i,t}}$ without knowing $h = g_1^{ab}$. Remark that ciphertexts $\{c_{i,t}\}$ must be decryptable by the adversary, i.e., the discrete logarithm $\log_{g_1} V_t$ must be sufficiently small. If all $x_{i,t}$ are related to $1/a$ as above, then $\log_{g_1} V_{t^*} = (\sum_{i=1}^n x'_{i,t})/a$ is not computable. Thus, for relatively small X'_t, we set $x_{i,t} := x'_{i,t}/a$ for $i \in [1, n-1]$ and set $x_{n,t} := X'_t - \sum_{i=1}^{n-1} x'_{i,t}/a$. Then, $\sum_{i=1}^n x_{i,t} = X'_t$ holds and $\log_{g_1} V_t = X'_t$ is computable by the adversary as in the scheme. For simulation, we need to decide each X'_t in the setup phase, and embed it to $v_{n,t}$ for utilizing the Boneh-Boyen technique. This is the reason why our scheme is weak aggregator unforgeable ($x_{i,t}$ is chosen by the oracle), and the size of verification keys linearly depend on t_{max}. Remark that we can achieve a tight reduction (i.e., $O(1)$ reduction loss) from the advantage of the mCDH problem, and this proof also works well for Type I forgery (simply we assume that the encryption oracle at t^* is not sent from $\mathcal{A}$, choose X'_{t^*} randomly, and $X_{t^*} \neq X'_{t^*}$ holds with overwhelming probability $1 - 1/p$).

Proof:

Type I Forgery : Let $(p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, (g_1^a, g_1^b, g_2^a))$ be an instance of the CDH problem. We construct an algorithm $\mathcal{B}$ that computes g_1^{ab} by using an adversary $\mathcal{A}$ that breaks weak aggregator unforgeability of our scheme as follows. $\mathcal{B}$ sets $\mathsf{param} = (p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, chooses γ, s_i, t_i, and sk_A as usual, chooses $v'_{i,t} \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [1, n]$ such that $\sum_{i=1}^n v'_{i,t} \neq 0$, and chooses $t \in [1, t_{\mathsf{max}}]$, and implicitly sets $v_{i,t} := v'_{i,t} a$. $\mathcal{B}$ computes $\mathsf{vk}_t = (g_2^a)^{\sum_{i=1}^n v'_{i,t}}$. $\mathcal{B}$ sends $(\mathsf{params}, \mathsf{sk}_A, \mathsf{vk} = \{\mathsf{vk}_t\}_{t \in [1, t_{\mathsf{max}}]})$ to $\mathcal{A}$.

Moreover, $\mathcal{B}$ guesses t^* (with success probability $1/t_{\mathsf{max}}$). For a time t, $\mathcal{B}$ chooses $\tilde{t} \xleftarrow{\$} \mathbb{Z}_p$ and sets $H_5(t)$ as

$$H_5(t) = \begin{cases} g_1^{\tilde{t}} & (t \neq t^*) \\ (g_1^b)^{\tilde{t}^*} & (t = t^*) \end{cases}$$

For other hash functions, $\mathcal{B}$ just returns a random value. For responding an encryption query (i, t) where $t \neq t^*$, $\mathcal{B}$ chooses $x_{i,t}$ and computes $c_{i,t}$ as usual, and computes $\sigma_{i,t} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} (g_1^a)^{v'_{i,t}\tilde{t}} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} (g_1^{\tilde{t}})^{a v'_{i,t}} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}}$. Remark that $\mathcal{A}$ does not send an encryption query at time t^* in this type.

Finally, at time t^*, $\mathcal{A}$ outputs (σ_{t^*}, X_{t^*}). From the verification equation, (σ_{t^*}, X_{t^*}) must satisfy $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^n v_{i,t}} h^{X_{t^*}}$. That is,

$$\sigma_{t^*} h^{-X_{t^*}} = H_5(t^*)^{\sum_{i=1}^n v_{i,t^*}} = ((g_1^b)^{\tilde{t}^*})^{a \sum_{i=1}^n v'_{i,t^*}}$$

holds. $\mathcal{B}$ solves the CDH problem by computing $(\sigma_{t^*} h^{-X_{t^*}})^{1/\tilde{t}^* \sum_{i=1}^n v'_{i,t^*}} = g_1^{ab}$.

Type II Forgery : Let $(p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, (g_1^a, g_1^b, g_1^{1/a}, g_2^a))$ be an instance of the Modified CDH problem. We construct an algorithm $\mathcal{B}$ that computes g_1^{ab} by using an adversary $\mathcal{A}$ that breaks weak aggregator unforgeability of our scheme as follows. $\mathcal{B}$ sets $\mathsf{param} = (p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, chooses γ, s_i, t_i, sk_A, and $v_{i,t}$ for $i = [1, n-1]$ and $t \in [1, t_{\mathsf{max}}]$ as usual. For $t \in [1, t_{\mathsf{max}}]$, $\mathcal{B}$ chooses $v'_{n,t} \xleftarrow{\$} \mathbb{Z}_p$, and also chooses $X'_t \xleftarrow{\$} \mathbb{Z}_p$ such that the size of X'_t is sufficiently small where the discrete logarithm problem $g_1^{X_t}$ with respect to basis g_1 can be solved. This is the necessary condition that ciphertexts can be decrypted by the adversary as in the scheme. For $t \in [1, t_{\mathsf{max}}]$, $\mathcal{B}$ chooses $\tilde{t} \xleftarrow{\$} \mathbb{Z}_p$ and sets $H_5(t)$ as $(g_1^b)^{\tilde{t}}$. $\mathcal{B}$ implicitly sets $v_{n,t} = v'_{n,t} + (-aX'_t)/\tilde{t}$. $\mathcal{B}$ computes $\mathsf{vk}_t = (g_2^a)^{-X'_t/\tilde{t}} g_2^{v'_{n,t} + \sum_{i=1}^{n-1} v_{i,t}}$. $\mathcal{B}$ implicitly sets $h = g_1^{ab}$ and computes $Z = e(g_1^b, g_2^a) = e(h, g_2)$. $\mathcal{B}$ sends $(\mathsf{params}, \mathsf{sk}_A, \mathsf{vk} = \{\mathsf{vk}_t\}_{t \in [1, t_{\mathsf{max}}]})$ to $\mathcal{A}$.

For responding an encryption query (i, t), $\mathcal{B}$ computes $(c_{i,t}, \sigma_{i,t})$ as follows. $\mathcal{B}$ chooses $x'_{i,t} \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [1, n-1]$ and implicitly sets $x_{i,t}$ as

$$x_{i,t} = \begin{cases} x'_{i,t}/a & (i \in [1, n-1]) \\ X'_t - \sum_{i=1}^{n-1} x'_{i,t}/a & (i = n) \end{cases}$$

and computes

$$\begin{aligned} i \in [1, n-1] : \ c_{i,t} &= (g_1^{1/a})^{x'_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \\ &= g_1^{x'_{i,t}/a} H_1(t)^{s_i} H_2(t)^{t_i} = g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \\ i = n : \ c_{i,t} &= g_1^{X'_t} (g_1^{1/a})^{-\sum_{i=1}^{n-1} x'_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \\ &= g_1^{X'_t - \sum_{i=1}^{n-1} x'_{i,t}/a} H_1(t)^{s_i} H_2(t)^{t_i} \\ &= g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \end{aligned}$$

and

$$\begin{aligned} i \in [1, n-1] : \ \sigma_{i,t} &= (g_1^b)^{x'_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}} \\ &= (g_1^{ab})^{x'_{i,t}/a} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}} \\ &= h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}} \\ i = n : \ \sigma_{i,t} &= (g_1^b)^{-\sum_{i=1}^{n-1} x'_{i,t} + \tilde{t} v'_{n,t}} H_3(t)^{s_i} H_4(t)^{t_i} \\ &= (g_1^{ab})^{X'_t} (g_1^b)^{-\sum_{i=1}^{n-1} x'_{i,t}} (g_1^{-ab})^{X'_t} (g_1^b)^{\tilde{t} v'_{n,t}} H_3(t)^{s_i} H_4(t)^{t_i} \\ &= (g_1^{ab})^{X'_t} (g_1^{ab})^{-\sum_{i=1}^{n-1} x'_{i,t}/a} (g_1^{-ab})^{X'_t} (g_1^b)^{\tilde{t} v'_{n,t}} H_3(t)^{s_i} H_4(t)^{t_i} \\ &= (g_1^{ab})^{X'_t - \sum_{i=1}^{n-1} x'_{i,t}/a} H_3(t)^{s_i} H_4(t)^{t_i} ((g_1^b)^{\tilde{t}})^{v'_{n,t} + (-aX'_t)/\tilde{t}} \\ &= h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}} \end{aligned}$$

Remark that $\sum_{i=1}^n x_{i,t} = X'_t$ and $\{c_{i,t}\}_{i \in [1,n]}$ can be decrypted by the adversary who has sk_A.

Finally, $\mathcal{A}$ outputs $(t^*, X_{t^*}, \sigma_{t^*})$ where $t^* \in [1, t_{\mathsf{max}}]$ and $X_{t^*} \neq X'_{t^*}$. From the verification equation, (σ_{t^*}, X_{t^*}) must satisfy $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^n v_{i,t^*}} h^{X_{t^*}}$. Here, $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^n v_{i,t^*}} h^{X_{t^*}} = ((g_1^b)^{\tilde{t^*}})^{v'_{n,t^*} + (-aX'_{t^*}/\tilde{t^*}) + \sum_{i=1}^{n-1} v_{i,t^*}} (g_1^{ab})^{X_{t^*}} = (g_1^{ab})^{X_{t^*} - X'_{t^*}} (g_1^b)^{\tilde{t^*}(v'_{n,t^*} + \sum_{i=1}^{n-1} v_{i,t^*})}$ holds. $\mathcal{B}$ computes

$$(\sigma_{t^*} / (g_1^b)^{\tilde{t^*}(v'_{n,t^*} + \sum_{i=1}^{n-1} v_{i,t^*})})^{1/(X_{t^*} - X'_{t^*})} = g_1^{ab}$$

and solves the mCDH problem. □

4.3 The Proposed Scheme 2: Providing Semi-Adaptive Aggregator Unforgeability

In the first scheme, $v_{i,t}$ is chosen in the setup phase. This leads to large-size verification keys, and is the reason why the first scheme provides weak aggregator unforgeability. As mentioned before, as another choice, a user i chooses $v_{i,t} \xleftarrow{\$} \mathbb{Z}_p$ at time t on the fly (i.e., in the encryption phase), computes

$\mathsf{vk}_{i,t} := g_1^{v_{i,t}}$, and sends $\mathsf{vk}_{i,t}$ to the aggregator together with $(c_{i,t}, \sigma_{i,t})$. Then $\mathsf{vk}_t = \prod_{i=1}^n \mathsf{vk}_{i,t}$ is used in the VerifySum algorithm. In this case, X_t, chosen by the adversary in the security proof, can be embedded to vk_t on the fly in the encryption oracle. Moreover, one hash function H and vk can be removed from the public value, and v_i can also be removed from sk_i.

One problem with this strategy is that the Enc algorithm becomes probabilistic. That is, a user is required to generate a random number $v_{i,t}$ for each time t. This could be problematic if users have limited computational power. Another problem is that the aggregator (which is an adversary of the aggregator unforgeability game) could modify vk_t, and the VerifySum algorithm is run by a maliciously generated vk_t. Then, no security is guaranteed. One solution is to use a bulletin board [26] which is publicly readable and every user can write to, but nobody can delete from. The bulletin board can be considered a public channel with memory. That is, a user i writes $\mathsf{vk}_{i,t}$ to the bulletin board BB. Remark that the computation cost of $\mathsf{vk}_t = \prod_{i=1}^n \mathsf{vk}_{i,t}$ is almost similar to that of $V_t = H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^n c_{i,t}$. That is, if a data analyzer who runs the VerifySum algorithm computes vk_t, then the data analyzer does not need to delegate the computation of the aggregated sum to the aggregator, and this leads to a wag-the-dog situation. So, we assume that the aggregator computes vk_t, and $\mathsf{vk}_{i,t}$ written in BB acts as a deterrent against the aggregator that modifies vk_t, since the data analyzer can check anytime whether vk_t provided by the aggregator is computed by $\{\mathsf{vk}_{i,t}\}_{i=1}^n$ or not. In summary, we slightly modify the syntax such that the bulletin board BB is added as an input of the Enc algorithm, and the AggrDec algorithm outputs vk_t together with (X_t, σ_t).

We give the second scheme as follows.

$\mathsf{Setup}(1^\lambda)$**:** Choose $(p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of λ-bit prime order $p = M$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. Let $H_i : \mathbb{Z} \to \mathbb{G}_1$ $(i = 1, 2, 3, 4, 5)$ be hash functions. Choose $\gamma, s_1, \ldots, s_n, t_1, \ldots, t_n \xleftarrow{\$} \mathbb{Z}_p$, set $s_0 = -\sum_{i=1}^n s_i$, $t_0 = -\sum_{i=1}^n t_i$, $h = g_1^\gamma$, and $Z = e(h, g_2)$. Output $\mathsf{param} = ((p, e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), Z, H_1, H_2, H_3, H_4, H_5)$, $\mathsf{sk}_A = (s_0, t_0)$, $\mathsf{sk}_i = (s_i, t_i, h)$, and $\mathsf{vk} = \emptyset$.

$\mathsf{Enc}(\mathsf{param}, t, x_{i,t}, \mathsf{sk}_i, \mathsf{BB})$**:** Parse $\mathsf{sk}_i = (s_i, t_i, h)$. Choose $v_{i,t} \xleftarrow{\$} \mathbb{Z}_p$, compute $\mathsf{vk}_{i,t} := g_1^{v_{i,t}}$, and compute

$$c_{i,t} = g_1^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \text{ and } \sigma_{i,t} = h^{x_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}}$$

and output $(c_{i,t}, \sigma_{i,t}, \mathsf{vk}_{i,t})$. Moreover, write $\mathsf{vk}_{i,t}$ to the bulletin board BB.

$\mathsf{AggrDec}(\mathsf{param}, t, \{(c_{i,t}, \sigma_{i,t}, \mathsf{vk}_{i,t})\}_{i=1}^n, \mathsf{sk}_A)$**:** Parse $\mathsf{sk}_A = (s_0, t_0)$. Compute

$$V_t = H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^n c_{i,t} = g_1^{X_t}$$

where $X_t = \sum_{i=1}^n x_{i,t}$, and solve the discrete logarithm V_t with respect to basis g_1. Moreover, compute

$$\sigma_t = H_3(t)^{s_0} H_4(t)^{t_0} \prod_{i=1}^n \sigma_{i,t} \text{ and } \mathsf{vk}_t = \prod_{i=1}^n \mathsf{vk}_{i,t}$$

Output $(X_t, \sigma_t, \mathsf{vk}_t)$.

$\mathsf{VerifySum}(\mathsf{param}, t, X_t, \sigma_t, \mathsf{vk}_t)$**:** Output 1 if

$$\frac{e(\sigma_t, g_2)}{e(H_5(t), \mathsf{vk}_t)} = Z^{X_t}$$

holds. Otherwise, output 0.

Theorem 4.3 *Our scheme 2 is aggregator obliviousness under the DDH assumption on $\mathbb{G}_1$ in the random oracle model.*

This is essentially the same as that of the first scheme. We omit it.

Theorem 4.4 *Our scheme 2 is semi-adaptively aggregator unforgeable under the DDH and mCDH assumptions in the random oracle model.*

Proof: The simulation is almost similar to that of the first scheme. Remark that $c_{i,t}$ is not a ciphertext of $x_{i,t}$ in the simulation. We can regard $c_{i,t}$ is a ciphertext of $r_{i,t}$ for some random $r_{i,t} \in \mathbb{Z}_p$, and $c_{n,t}$ is a ciphertext of $\sum_{i=1}^{n} x_{i,t} - \sum_{i=1}^{n-1} r_{i,t}$. Thus, we need to show that these modifications do not affect the security. We reduce the indistinguishability to aggregator obliviousness as follows. We define sequential of games. Let Game_0 be the original game, and Game_1 be the same as Game_0 except that ciphertexts and tags are computed as above. We define subgames Game_j for $j \in [1, t_{\mathsf{max}} + 1]$ where $\mathsf{Game}_{t_{\mathsf{max}}} := \mathsf{Game}_1$. Let $\mathcal{C}$ be the challenger of aggregator obliviousness that prepares $(\mathsf{param}, \mathsf{sk}_A, \{\mathsf{sk}_i\}_{i=1}^{n})$, and $\mathcal{B}$ be the simulator. $\mathcal{B}$ requests sk_A to $\mathcal{C}$, and sends $(\mathsf{param}, \mathsf{sk}_A)$ to the adversary $\mathcal{A}$. Let $\{(i, j, x_{i,j})\}_{i=1}^{n}$ be the j-th encryption query. $\mathcal{B}$ randomly chooses $r_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [1, n-1]$, and sets the challenge message $(\mathbb{U}, j := t^*, \{(x_{i,j}^{(0)}, x_{i,j}^{(1)})\}_{i \in \mathbb{U}})$ where $x_{i,j}^{(0)} = x_{i,j}$ for $i \in [1, n]$, $x_{i,j}^{(1)} = r_{i,j}$ for $i \in [1, n-1]$, and $x_{n,j}^{(1)} = \sum_{i=1}^{n} x_{i,j} - \sum_{i=1}^{n-1} r_{i,j}$. Remark that $\sum_{i=1}^{n} x_{i,j}^{(0)} = \sum_{i=1}^{n} x_{i,j}^{(1)}$ holds. If $b = 0$, then it simulates Game_{j-1}, and if $b = 1$, then it simulates Game_j. Thus, two games are indistinguishable due to aggregator obliviousness. This modifications require $O(t_{\mathsf{max}})$ reduction loss, and require $O(t_{\mathsf{max}}^2)$ reduction loss from the advantage of the DDH problem.

In $\mathsf{Game}_{t_{\mathsf{max}}} := \mathsf{Game}_1$, for each encryption query $\{(i_n, t, x_{i_n,t})\}_{i=1}^{n}$, choose $x'_{i,t} \xleftarrow{\$} \mathbb{Z}_p$ (regardless of $x_{i,t}$) and $v_{i,t} \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [1, n-1]$, compute

$$c_{i,t} = (g_1^{1/a})^{x'_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \text{ and } \sigma_{i,t} = (g_1^b)^{x'_{i,t}} H_3(t)^{s_i} H_4(t)^{t_i} H_5(t)^{v_{i,t}}$$

For $i = n$, choose $v'_{i,t} \xleftarrow{\$} \mathbb{Z}_p$, compute $X'_t = \sum_{i=1}^{n} x_{i,t}$ from queries $\{(i, t, x_{i,t})\}_{i \in [1,n]}$, and compute

$$c_{i,t} = g_1^{X'_t} (g_1^{1/a})^{-\sum_{i=1}^{n-1} x'_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \text{ and } \sigma_{i,t} = (g_1^b)^{-\sum_{i=1}^{n-1} x'_{i,t} + \tilde{t} v'_{n,t}} H_3(t)^{s_i} H_4(t)^{t_i}$$

Here, $H_5(t)$ is set as $(g_1^b)^{\tilde{t}}$ as in the proof of the first scheme. Return $\{(c_{i,t}, \sigma_{i,t}, \mathsf{vk}_{i,t})\}_{i=1}^{n}$ to $\mathcal{A}$, and write $\mathsf{vk}_{i,t} = g_2^{v_{i,t}}$ for $i \in [1, n-1]$ and $\mathsf{vk}_{i,t} = (g_2^a)^{-X'_t/\tilde{t}} g_2^{v'_{n,t}}$ to BB. We note that $\{c_{i,t}\}_{i \in [1,n]}$ can be decrypted by the adversary, and the decryption result is exactly $\sum_{i=1}^{n} x_{i,t}$ that the adversary queried.

Finally, $\mathcal{A}$ outputs $(t^*, X_{t^*}, \sigma_{t^*})$ where $t^* \in [1, t_{\mathsf{max}}]$ and $X_{t^*} \neq X'_{t^*}$. From the verification equation, (σ_{t^*}, X_{t^*}) must satisfy $\sigma_{t^*} = H_5(t^*)^{\sum_{i=1}^{n} v_{i,t^*}} h^{X_{t^*}}$. $\mathcal{B}$ computes

$$(\sigma_{t^*} / (g_1^b)^{\tilde{t^*}(v'_{n,t^*} + \sum_{i=1}^{n-1} v_{i,t^*})})^{1/(X_{t^*} - X'_{t^*})} = g_1^{ab}$$

and solves the mCDH problem. □

5 Regarding Message Space

As in the definition of previous works [9, 49, 36], for some fixed integer M, we assume that $x_{i,t} \in \mathbb{Z}_M$ and aggregator obliviousness requires the condition $\sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(0)} \bmod M = \sum_{i \in \mathbb{S}_{t^*}} x_{i,t^*}^{(1)} \bmod M$ (when $\mathbb{S}_{t^*} = \mathbb{U}$ and sk_A is compromised). Since $M = p$, $x_{i,t}$ might be a large value even its summation is required to be sufficiently small. In the definitions of aggregator obliviousness and full/semi-adaptive aggregator unforgeability, an adversary chooses $x_{i,t}$, and thus selecting such a large $x_{i,t}$ is acceptable (since this is just a strategy of the adversary). On the other hand, in the definition of weak aggregator unforgeability, the encryption oracle *randomly* chooses $x_{i,t}$ from $\mathbb{Z}_M$. If it is desirable to restrict $x_{i,t}$ to be small, then we can modify the definition of the encryption oracle such that the encryption oracle randomly chooses $x_{i,t}$ from a small message space. Then, each $x_{i,t}$ is not rounded up when its summation is computed by modulo p. Remark that this modification requires an additional reduction loss $O(t_{\mathsf{max}})$. In the security proof of the weak aggregator unforgeability, a part of mCDH instance a is embedded into $x_{i,t}$, and thus $x_{i,t}$ is a random value of $\mathbb{Z}_p$. So, as in the security proof of the second scheme, we need to define sequential of games, and replace ciphertexts and tags of $x_{i,t}$ to those of $r_{i,t} \in \mathbb{Z}_p$ or $\sum_{i=1}^{n} x_{i,t} - \sum_{i=1}^{n-1} r_{i,t}$. This requires $O(t_{\mathsf{max}})$ reduction loss, and requires $O(t_{\mathsf{max}}^2)$ reduction loss from the advantage of the DDH problem in total.

6 Conclusion and Open Problem

In this paper, we propose two aggregator oblivious encryption schemes with public verifiability from static and simple assumptions. The first scheme just provides weak aggregator unforgeability, and it seems still meaningful in the smart meter settings since power consumption is measured by the meter. Though the scheme requires $O(t_{\mathsf{max}})$-size verification keys, and it could be a bottleneck for supporting long-term period, the scheme still efficiently works for a relatively short-term period. The second scheme provides semi-adaptive aggregator unforgeability and constant-size verification keys, whereas we need to additionally assume the existence of public channels with memory, such as bulletin board [26]. Thus, proposing aggregator oblivious encryption scheme providing full aggregator unforgeability from simple and static assumptions is still open problem. In our schemes, a value h is shared by all users as their secret key. The value has a crucial role for providing unforgeability. Obviously, if h is revealed, then anyone can easily produce a forged tag. That is, our scheme is vulnerable against the corruption attack where an adversary (modeled as a malicious aggregator) obtains secret keys of corrupted users. So, it is desirable to provide unforgeability with the collusion resistance. Moreover, as in [9], proposing a generic construction of aggregator oblivious encryption with public verifiability (containing a Paillier-type instantiation) also could be an interesting open problem. Since we consider summation as the aggregation function, constructing (verifiable) aggregator oblivious encryption with rich aggregation functions from (verifiable) multi-input functional encryption [1, 4, 24, 12] also could be an interesting open problem.

Acknowledgement: The author would like to thank Dr. Miyako Ohkubo for her invaluable comments against the bulletin board assumption, thank Dr. Takuya Hayashi for his invaluable suggestions against elliptic curve parameters, and thank Dr. Takahiro Matsuda for his invaluable comments against the size of message space. This work was partially supported by JSPS KAKENHI Grant Number JP16K00198.

References

[1] M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT*, pages 601–626, 2017.

[2] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *J. Cryptology*, 29(4):833–878, 2016.

[3] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *ACM CCS*, pages 863–874, 2013.

[4] S. Badrinarayanan, V. Goyal, A. Jain, and A. Sahai. Verifiable functional encryption. In *ASIACRYPT*, pages 557–587, 2016.

[5] E. Barker. NIST Special Publication 800-57 Part 1, Revision 4. Available at `http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4`, January 2016.

[6] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[7] G. Barthe, G. Danezis, B. Grégoire, C. Kunz, and S. Z. Béguelin. Verified computational differential privacy with applications to smart metering. In *IEEE Computer Security Foundations Symposium*, pages 287–301, 2013.

[8] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *EUROCRYPT*, pages 399–416, 1996.

[9] F. Benhamouda, M. Joye, and B. Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*, 18(3):10, 2016.

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[11] X. Boyen. The Uber-assumption family. In *Pairing-Based Cryptography*, pages 39–56, 2008.

[12] Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *EUROCRYPT*, pages 852–880, 2016.

[13] T. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*, pages 200–214, 2012.

[14] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, pages 103–118, 1997.

[15] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[16] Y. Cui, E. Fujisaki, G. Hanaoka, H. Imai, and R. Zhang. Formal security treatments for IBE-to-signature transformation: Relations among security notions. *IEICE Transactions*, 92-A(1):53–66, 2009.

[17] G. Danezis, C. Fournet, M. Kohlweiss, and S. Z. Béguelin. Smart meter aggregation via secret-sharing. In *ACM Workshop on Smart Energy Grid Security*, pages 75–80, 2013.

[18] A. Datta and M. Joye. Cryptanalysis of a privacy-preserving aggregation protocol. *IEEE Trans. Dependable Sec. Comput.*, 2017, to appear.

[19] K. Emura. Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions. In *ACISP*, 2017, to appear.

[20] C. Fan, S. Huang, and Y. Lai. Privacy-enhanced data aggregation scheme against internal attackers in smart grid. *IEEE Trans. Industrial Informatics*, 10(1):666–675, 2014.

[21] D. Fiore, R. Gennaro, and V. Pastro. Efficiently verifiable computation on encrypted data. In *ACM CCS*, pages 844–855, 2014.

[22] F. D. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In *Security and Trust Management*, pages 226–238, 2010.

[23] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[24] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F. Liu, A. Sahai, E. Shi, and H. Zhou. Multi-input functional encryption. In *EUROCRYPT*, pages 578–602, 2014.

[25] M. Green and S. Hohenberger. Practical adaptive oblivious transfer from simple assumptions. In *TCC*, pages 347–363, 2011.

[26] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT*, pages 539–556, 2000.

[27] D. Hofheinz and T. Jager. Verifiable random functions from standard assumptions. In *TCC-A*, pages 336–362, 2016.

[28] T. Jager. Verifiable random functions from weaker assumptions. In *TCC*, pages 121–143, 2015.

[29] M. Jawurek, M. Johns, and F. Kerschbaum. Plug-in privacy for smart metering billing. In *Privacy Enhancing Technologies*, pages 192–210, 2011.

[30] M. Jawurek and F. Kerschbaum. Fault-tolerant privacy-preserving statistics. In *Privacy Enhancing Technologies*, pages 221–238, 2012.

[31] M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography*, pages 111–125, 2013.

[32] T. Jung, X. Li, and M. Wan. Collusion-tolerable privacy-preserving sum and product calculation without secure channel. *IEEE Trans. Dependable Sec. Comput.*, 12(1):45–57, 2015.

[33] E. Kiltz and Y. Vahlis. CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption. In *CT-RSA*, pages 221–238, 2008.

[34] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *CRYPTO*, pages 543–571, 2016.

[35] I. Leontiadis, K. Elkhiyaoui, and R. Molva. Private and dynamic time-series data aggregation with trust relaxation. In *CANS*, pages 305–320, 2014.

[36] I. Leontiadis, K. Elkhiyaoui, M. Önen, and R. Molva. PUDA - privacy and unforgeability for data aggregation. In *CANS*, pages 3–18, 2015.

[37] Q. Li and G. Cao. Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error. In *Privacy Enhancing Technologies*, pages 60–81, 2013.

[38] B. Libert, F. Mouhartem, T. Peters, and M. Yung. Practical "signatures with efficient protocols" from simple assumptions. In *AsiaCCS*, pages 511–522, 2016.

[39] B. Libert, T. Peters, and M. Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *CRYPTO*, pages 296–316, 2015.

[40] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen. EPPA: an efficient and privacy-preserving aggregation scheme for secure smart grid communications. *IEEE Trans. Parallel Distrib. Syst.*, 23(9):1621–1631, 2012.

[41] A. Menezes, P. Sarkar, and S. Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. *IACR Cryptology ePrint Archive*, 2016:1102, 2016.

[42] S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. *J. Cryptology*, 15(1):1–18, 2002.

[43] M. Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.

[44] K. Ohara, Y. Sakai, F. Yoshida, M. Iwamoto, and K. Ohta. Privacy-preserving smart metering with verifiability for both billing and energy management. In *ASIAPKC*, pages 23–32, 2014.

[45] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[46] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[47] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *ACM SIGMOD*, pages 735–746, 2010.

[48] A. Rial and G. Danezis. Privacy-preserving smart metering. In *WPES*, pages 49–60, 2011.

[49] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.

[50] K. Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In *SCN*, pages 298–317, 2014.

[51] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.