A New Framework for Privacy-Preserving Aggregation of Time-Series Data

Fabrice Benhamouda, Marc Joye, Benoit Libert

A New Framework for Privacy-Preserving Aggregation of Time-Series Data

FABRICE BENHAMOUDA, ENS, CNRS, INRIA, and PSL, Paris, France
MARC JOYE, Technicolor, Los Altos, USA
BENOÎT LIBERT, ENS Lyon, Lyon, France

Aggregator-oblivious encryption is a useful notion put forward by Shi *et al.* in 2011 that allows an untrusted aggregator to periodically compute an aggregate value over encrypted data contributed by a set of users. Such encryption schemes find numerous applications, in particular in the context of privacy-preserving smart metering.

This paper presents a general framework for constructing privacy-preserving aggregator-oblivious encryption schemes using a variant of Cramer-Shoup's paradigm of smooth projective hashing. This abstraction leads to new schemes based on a variety of complexity assumptions. It also improves upon existing constructions, providing schemes with shorter ciphertexts and better encryption times.

General Terms: Data aggregation, Privacy, Cryptography

Additional Key Words and Phrases: Private aggregation, aggregator-oblivious encryption, smart metering, smooth projective hashing, security reductions

1. INTRODUCTION

Since the introduction of electricity into the home, its usage has been recorded by an electricity meter attached to the exterior of the homes. This situation is however gradually changing with the progressive deployment of smart meters [McDaniel and McLaughlin 2009]. In addition to the basic service offered by its predecessor, a smart meter comes with extra useful features aiming at reducing energy costs. For example, a smart meter can turn down momentarily high-energy electrical appliances during peak hours. For the utility company, one of its most appealing features is its ability to report in almost real-time the power consumption of their consumers. This fine-grained information is very helpful as it allows the electricity provider to better adapt the load or forecast the supply. Moreover, it allows the electricity provider to quickly react when anomalies are detected on the grid. The resulting savings are also beneficial to the consumers as they give rise to better pricing. But there is a downside. Frequent usage reports leak information about the consumer habits and behaviors —for example, when a certain consumer turns her TV on and what programs she is likely to watch. These seemingly unimportant issues should not be underestimated as they may have unintended consequences from the inference of some private information (attributes or data).

In most cases, there is no need for the utility company (except for preparing the monthly bill) to get the fined-grained energy usage of *each* customer. For example, in the previous scenario, an aggregate suffices. The goal of this paper is to mitigate the privacy issues that arise from smart metering by computing aggregates rather than individual energy consumption. More generally, we are seeking efficient privacy-preserving methods for the aggregation of time-series data. The entity computing the aggregates is not necessarily trusted. We are interested in solutions that affect the existing infrastructure as little as possible. In particular, in the above scenario, we do not require smart meters to interact with each other nor the existence of a return channel. We note that the billing issue is separate. In practice, smart meters report separately their monthly energy consumption to the energy provider.

Related work. The above setting is the one considered in [Shi et al. 2011] and [Joye and Libert 2013]. Each smart meter encrypts its actual energy consumption and sends the result at regular intervals to an aggregator (that can be an entity different from the

energy provider). In the case of electricity metering, a typical time period is 15 minutes. Upon receiving the encrypted values from a predetermined set of users, the aggregator combines the received values and therefrom deduces the total energy consumption over the population of these users for the current time period. This operation involves a secret key known only to the aggregator. Further, computing the sum over the predetermined set of users is the only operation the aggregator can perform —it cannot learn anything beyond what is revealed by the aggregate value. Following [Shi et al. 2011], such a scheme is termed an *aggregator-oblivious encryption scheme*.

Like [Shi et al. 2011] and [Joye and Libert 2013], all our schemes can serve as a building block for the fault tolerant solution of [Chan et al. 2012] while enjoying the benefits of our construction. In fact, all the extensions of [Shi et al. 2011] are also possible with our system. In particular, although the focus of this paper is put on the encryption, the proposed schemes are compatible with the differential-privacy framework [Dwork 2008; Dwork et al. 2006]. In this case, the smart meters simply need to add some appropriately-generated noise to their data prior to encryption

Two other protocols in settings similar to the one of aggregator-oblivious encryption are the low-overhead protocol of Kursawe et al. [2011] and the protocol of Ács and Castelluccia [2011]. These protocols however have the drawback of requiring each smart meter to store n keys, which can be impractical for a large set of smart meters.

We also note that recent results on multi-input functional encryption [Goldwasser et al. 2014] imply non-interactive constructions of aggregator-oblivious encryption. However, due their inevitable use of indistinguishability obfuscation candidates [Garg et al. 2013], they are really far from being practical.

Many other settings than the one we consider have been studied in the litterature. For example, the protocol of Dwork et al. in [Dwork et al. 2006] allows the aggregation of more complex functions, but requires communication between the smart meters. The protocols of Rastogi and Nath [Rastogi and Nath 2010] and of Garcia and Jacobs [Garcia and Jacobs 2010] require bi-directional channels between the smart meters and the aggregator, while we only require a uni-directional channel from each smart meter to the aggregator. Leontiadis, Elkhiyaoui, and Moval [Leontiadis et al. 2014], as well as Kawurek and Kerschbaum [Jawurek and Kerschbaum 2012] suppose the existence of an additional semi-trusted party, who cannot collude with the aggregator. In addition, in the second paper, the smart meter should be able to receive data from this third party. For more information on aggregation schemes, we refer the reader to the detailed survey of Kawurek, Kerschbaum and Danezis [Jawurek et al. 2012].

Our contributions. While applicable to our framework, the solutions offered in [Shi et al. 2011] and [Joye and Libert 2013] are not fully satisfying, but for different reasons. Table 1 gives a rough idea of the expected gains for our basic aggregator-oblivious encryption scheme (a more detailed analysis with concrete implementation numbers is provided in Section 3).

Joye-Libert's scheme supports large plaintext spaces and a large number of users. However as it is built over Paillier's encryption scheme, the involved parameters are somewhat large. Back to our example of smart meters, this in turn implies that these are likely equipped with crypto-processors for modular arithmetic over large integers and possess sufficient memory for storing intermediate computations. Larger ciphertexts also mean more bandwidth for their transmission. Shi *et al.*'s scheme provides a cheaper solution as it is ElGamal-based and relies on the Decisional Diffie-Hellman assumption (DDH). In particular, it can be implemented using elliptic curve groups with much shorter parameters. It requires the computed aggregated sum to lie in a relatively small predetermined set. In the case of smart meters, this does not really constitute a limitation for most practical settings, as the sum should be less than 30 bits long.

Table I. Reduction loss and typical parameter size for existing schemes and our basic scheme (for $T = n = 2^{20}$)

Security level	Scheme	Reduction loss	Typical size (bits)		
			Group elements	Private keys	Ciphertexts
80 bits	[Shi et al. 2011]	$\approx 2^{80}$	320	320	320
80 bits	[Joye and Libert 2013]	$\lesssim 2^{20}$	≤ 3862	≤ 3862	≤ 3862
80 bits	**This work**	$\lesssim \mathbf{2^{20}}$	$\mathbf{\leq 200}$	$\mathbf{\leq 400}$	$\mathbf{\leq 200}$
128 bits	[Shi et al. 2011]	$\approx 2^{80}$	416	416	416
128 bits	[Joye and Libert 2013]	$\lesssim 2^{20}$	≤ 8900	≤ 8900	≤ 8900
128 bits	**This work**	$\lesssim \mathbf{2^{20}}$	$\mathbf{\leq 296}$	$\mathbf{\leq 592}$	$\mathbf{\leq 296}$

But as already pointed out in [Joye and Libert 2013], one drawback of [Shi et al. 2011] is that the security reduction to the underlying complexity assumption is very loose. If the scheme is set up for n users and if the number of periods is at most T, there is a multiplicative gap of $\Theta(Tn^3)$ between the adversary's advantage and the reduction's probability to solve the DDH problem. The security loss factor is an important parameter as it defines the "exact" security [Bellare and Rogaway 1996] and thus helps in adequately selecting the key size. Unfortunately, using a meta-reduction, we show in Appendix C that a degradation factor of at least $\Omega(n^2)$ is unavoidable in the scheme of [Shi et al. 2011].

An important contribution of this paper is a new DDH-based aggregator oblivious encryption scheme (cf. Section 3) with a much tighter security reduction. While the security loss is of $O(Tn^3)$ in Shi *et al.*'s scheme, that in our basic scheme is roughly of at most $O(T)$ in the worst case scenario (this worst case scenario is illustrated by the figures given in Table 1).

In Section 4, we generalize our basic DDH-based construction. We propose a generic framework for the privacy-preserving aggregation of time-series data featuring a tighter reduction. This framework is based on smooth projective hash functions [Cramer and Shoup 2002] (SPHFs) with an additional property: key-homomorphism. As shown in Section 5, our framework encompasses our basic scheme as well as a variation of Joye-Libert's scheme. Several other aggregator-oblivious encryption schemes based on a variety of complexity assumptions are presented in Section 5. This clearly demonstrates the generic aspect of our framework.

2. AGGREGATOR-OBLIVIOUS ENCRYPTION

We review the definition of aggregator-oblivious (secret-key) encryption and then proceed with the corresponding security notion. We refer the reader to [Shi et al. 2011] for further introductory background.

Definition 2.1. An *aggregator-oblivious encryption scheme* is a tuple of three algorithms, $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggrDec})$, defined as:

$\mathsf{Setup}(1^\lambda)$. On input a security parameter λ, a trusted dealer generates the system parameters param, the aggregator's private key sk_0, and the private key sk_i for each user i $(1 \leq i \leq n)$.

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$. At time period t, user i encrypts a value $x_{i,t}$ using her private encryption key sk_i to get $c_{i,t} = \mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$.

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. At time period t, the aggregator using sk_0 obtains

$$X_t = \sum_{i=1}^{n} x_{i,t} \bmod M\,,$$

as the evaluation of $X_t = \mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. M is some fixed integer contained in the system parameters param.

This definition slightly generalizes the definition introduced in [Shi et al. 2011]. We make explicit the fact the sum is computed modulo some integer M. To compute sums over the integers, it is sufficient to choose M greater than the maximal possible sum, as done in the constructions of [Shi et al. 2011]. Furthermore, we note that some constructions assume $\mathsf{AggrDec}$ only works on a small subset of $\{0, \ldots, M-1\}$.

2.1. Aggregator obliviousness

Basically, the security notion of *aggregator obliviousness* (AO) requires that the aggregator cannot learn, for each time period, anything more than the aggregate value X_t from the encrypted values of n (honest) users. If there are corrupted users (i.e., users sharing their private information with the aggregator), the notion only requires that the aggregator gets no extra information about the values of the honest users beyond their aggregate value. Further, it is assumed that each user encrypts only one value per time period.

More formally, AO is defined by the following game between a challenger and an attacker.

Setup. The challenger runs the Setup algorithm and gives param to the attacker.
Queries. In a first phase, the attacker can submit queries that are answered by the challenger. The attacker can make two types of queries:
(1) Encryption queries: The attacker submits tuples $(i, t, x_{i,t})$ for a pair (i, t) and gets back the encryption of $x_{i,t}$ under key sk_i for time period t;
(2) Compromise queries: The attacker submits i and receives the private key sk_i of user i; if $i = 0$, the attacker receives the private key of the aggregator.
Challenge. In a second phase, the attacker chooses a time period $t^\star$. Let $\mathcal{U}^\star \subseteq \{1, \ldots, n\}$ be the whole set of users for which, at the end of the game, no encryption queries have been made on time period $t^\star$ and no compromise queries have been made. The attacker chooses a subset $\mathcal{S}^\star \subseteq \mathcal{U}^\star$ and two different series of triples

$$\langle (i, t^\star, x_{i,t^\star}^{(0)}) \rangle_{i \in \mathcal{S}^\star} \quad \text{and} \quad \langle (i, t^\star, x_{i,t^\star}^{(1)}) \rangle_{i \in \mathcal{S}^\star} ,$$

that are given to the challenger. Further, if the aggregator capability sk_0 is compromised at the end of the game and $\mathcal{S}^\star = \mathcal{U}^\star$, it is required that

$$\sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(0)} \bmod M = \sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(1)} \bmod M \ . \tag{1}$$

Guess. The challenger chooses at random a bit $b \in \{0, 1\}$ and returns the encryption of $\langle x_{i,t^\star}^{(b)} \rangle_{i \in \mathcal{S}^\star}$ to the attacker.
More queries. The attacker can make more encryption and compromise queries. Note that since $\mathcal{S}^\star \subseteq \mathcal{U}^\star$, the attacker cannot submit an encryption query $(i, t^\star, \cdot)$ with $i \in \mathcal{S}^\star$ or a compromise query i with $i \in \mathcal{S}^\star$.
Outcome. At the end of the game, the attacker outputs a bit b' and wins the game if and only if $b' = b$. As usual, $\mathcal{A}$'s advantage is defined to be

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) := 2\big|\Pr[b' = b] - 1/2\big| \ .$$

Definition 2.2. An encryption scheme is said to meet the AO *security notion* if no probabilistic polynomial-time attacker can guess correctly, in the above game, the bit b with a probability non-negligibly better (in the security parameter) than $1/2$. The probability is taken over the random coins of the game according to the distribution induced by Setup and over the random coins of the attacker.

2.2. Existing schemes

So far, there are two known constructions of AO encryption schemes. They both meet the AO security notion, in the random oracle model. The first one is due to Shi, Chan, Rieffel, Chow and Song [2011] and works in DDH groups. The second construction, due to Joye and Libert [2013], relies on the composite residuosity assumption [Paillier 1999]. These two schemes are reviewed in Appendix A.

3. A NEW DDH-BASED SCHEME

As aforementioned, the security proof offered in [Shi et al. 2011] incurs an $O(Tn^3)$ degradation factor. The scheme in [Joye and Libert 2013] avoids this degradation factor; namely, the multiplicative gap between the adversary's maximal advantage and the probability to break the underlying complexity assumption is only proportional to the number q_{enc} of encryption queries made by the adversary for distinct periods other than $t^\star$ (so that, $q_{enc} < T$). In this section, we introduce an aggregator-oblivious encryption scheme enjoying a security reduction as tight as in [Joye and Libert 2013] but based on the DDH assumption. The main advantage is that the resulting ciphertexts are much shorter.

3.1. Basic scheme

We base the security of our basic scheme on the standard DDH assumption.

Definition 3.1. Let $\mathbb{G}$ be a group of prime order p. The *Decision Diffie-Hellman (*DDH*)* problem in $\mathbb{G}$ is to distinguish among the following two distributions:

$$D_0 = \left\{(g, g^a, g^b, g^{ab}) \mid g \xleftarrow{R} \mathbb{G},\ a, b \xleftarrow{R} \mathbb{Z}_p\right\}$$

and

$$D_1 = \left\{(g, g^a, g^b, g^c) \mid g \xleftarrow{R} \mathbb{G},\ a, b, c \xleftarrow{R} \mathbb{Z}_p\right\} \ .$$

The DDH assumption states that the advantage of a polynomial-time distinguisher $\mathcal{A}$, defined as

$$\mathbf{Adv}^{\mathsf{DDH}}(\mathcal{A}) = \big|\Pr[\mathcal{A}(g,u,v,w) = 1 \mid (g,u,v,w) \xleftarrow{R} D_0] - \Pr[\mathcal{A}(g,u,v,w) = 1 \mid (g,u,v,w) \xleftarrow{R} D_1]\big|$$

is negligible.

We are now ready to present the scheme. It is given by the following tuple of algorithms.

$\mathsf{Setup}(1^\lambda)$. Let a group $\mathbb{G}$ of prime order $M = p$ for which the DDH assumption holds, and let a random generator $g \in \mathbb{G}$. Let also two hash functions $H_1 : \mathbb{Z} \to \mathbb{G}$ and $H_2 : \mathbb{Z} \to \mathbb{G}$. Finally, for $2n$ random elements $s_1, \ldots, s_n, t_1, \ldots, t_n \xleftarrow{R} \mathbb{Z}_p$, define $s_0 = -\sum_{i=1}^n s_i \bmod p$ and $t_0 = -\sum_{i=1}^n t_i \bmod p$.
The system parameters are $\mathsf{param} = \{p, \mathbb{G}, g, H_1, H_2\}$ and the secret key of user i is $\mathsf{sk}_i = (s_i, t_i)$, with $0 \le i \le n$.
$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$. At time period t, for a private input $x_{i,t} \in \mathbb{Z}_p$, user i produces

$$c_{i,t} = g^{x_{i,t}} H_1(t)^{s_i} H_2(t)^{t_i} \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. The aggregator obtains the sum $X_t = \sum_{i=1}^n x_{i,t}$ for time period t by first computing $V_t := H_1(t)^{s_0} H_2(t)^{t_0} \prod_{i=1}^n c_{i,t} = g^{X_t}$ and next the discrete logarithm of V_t w.r.t. basis g.

As for the Shi *et al.* construction, since g has order p, the sum X_t is computed modulo $M = p$. Further, in the AggrDec algorithm, the aggregator has to obtain the value of X_t from $V_t = g^{X_t}$ in $\mathbb{G}$. The most appropriate method for computing discrete logarithms is

Pollard's λ algorithm (or variants thereof) and requires that the range of X_t be relatively small.

3.2. Security

The next theorem proves that the basic scheme meets the AO security notion in the random oracle model, based on the DDH assumption.

THEOREM 3.2. *The scheme provides* AO *security under the* DDH *assumption in the random oracle model. Specifically, for any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a* DDH *distinguisher $\mathcal{B}_{\mathsf{DDH}}$ with comparable running time and such that*

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq 2e \cdot (q_{enc} + 1) \cdot (\mathbf{Adv}^{\mathsf{DDH}}(\mathcal{B}_{\mathsf{DDH}}) + T/p)\,,$$

where q_{enc} is the number of encryption queries made by the adversary for distinct periods other than $t^\star$, T is the maximum number of periods and e is the base for the natural logarithm.

PROOF. The theorem is a corollary of Theorem 4.3, which ensures the security of our abstract scheme (cf. Section 4.4). □

3.3. Performance

The new scheme combines the advantages of [Joye and Libert 2013] (which offers tighter security in the random oracle model) and of [Shi et al. 2011] (which has more compact ciphertexts when implemented using elliptic curve groups). As already shown in Table 1 (Section 1), our basic scheme represents the aggregator-oblivious encryption with the shortest ciphertexts. As exemplified in Table 3.3, it also features better encryption times.

Table 1 was constructed by taking $n = 2^{20} \approx 10^6$ users and $T = 2^{20}$ time periods. This approximately allows computing an aggregation every 15 minutes for 30 years over a city like Paris (there are about one million households in Paris). Moreover, to have the fairest possible comparison, the worst case for our reduction is considered: $q_{enc} = T - 1 \approx 2^{20}$. The key sizes are derived from the ECRYPT 2 recommendations [ECRYPT II 2012] — where Shi *et al.*'s scheme and our basic scheme are implemented in elliptic curve groups and Joye-Libert's scheme in $\mathbb{Z}_{N^2}^*$ with N an RSA modulus.

Table 3.3 presents the corresponding running times for a security level of 80 bits. Higher security levels yield better results for our basic scheme.

Table II. Running times (with margin of error at 95% confidence, computed with 100 samples) of our basic scheme and the existing schemes (for $T = n = 2^{20}$, 80-bit security, using parameters in Table 1, SHA-512 for hashing, 24-bit $x_{i,t}$, and X_t in a pre-determined 24-bit range, on an Intel™ Core i5 750 with MIRACL™ library https://github.com/CertiVox/MIRACL, Jun 20, 2013)

Scheme	Time (ms)			
	Hashing[a]	Encryption[b]	First phase of decryption[c]	Second phase of decryption[d]
[Shi et al. 2011]	0.23 (±0.01)	5.5 (±0.1)	11.3 (±0.0)	192 (±20)
[Joye and Libert 2013]	0.01 (±0.00)	58.3 (±0.5)	45.5 (±0.0)	0.0 (±0.0)
Our basic scheme	**0.23 (±0.01)**	**2.6 (±0.1)**	**6.9 (±0.0)**	**126 (±13)**

[a] Computation of $H(t)$ or $H_1(t)$ and $H_2(t)$;
[b] Computation of $c_{i,t}$, excluding computation of $H(t)/H_1(t)/H_2(t)$;
[c] Computation of V_t from $(c_{i,t})$, excluding computation of $H(t)/H_1(t)/H_2(t)$;
[d] Computation of X_t from V_t (we used a variant of the Pollard's kangaroo (or λ) method described in [Montenegro and Tetali 2009]).

4. GENERALIZATION USING KEY-HOMOMORPHIC SMOOTH PROJECTIVE HASH FUNCTIONS

In this section, we use the framework of key-homomorphic smooth projective hashing to generalize our DDH construction presented in Section 3.

4.1. Key-homomorphic smooth projective hash functions

Subset membership problem. We start with the important notion of subset membership problems, as introduced in [Cramer and Shoup 2002]. Consider an NP-language $\mathcal{L} \subset \mathcal{X}$, defined by a polynomial-time witness relation $\mathcal{R}$:

$$\mathcal{L} = \left\{ y \in \mathcal{X} \mid \exists w \text{ such that } \mathcal{R}(y, w) = 1 \right\} .$$

We suppose that $\mathcal{L}$, $\mathcal{X}$, $\bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$ are efficiently (uniformly) samplable, and even that sampling a word $y \in \mathcal{L}$ along with an associated witness w for this word can also be done efficiently. We also suppose that $|\mathcal{L}|/|\mathcal{X}|$ is negligible, or in other words that a random word in $\mathcal{X}$ is in $\bar{\mathcal{L}}$ with overwhelming probability.

Basically, the language $\mathcal{L}$ induces a hard subset membership problem if random elements of $\mathcal{L}$ cannot be distinguished from random elements of $\mathcal{X}$. More formally, this notion can be defined by the following game between a challenger and an attacker. The challenger chooses at random a bit b. The attacker can issue up to q_m (a parameter of the game) queries to the challenger. On each query, the challenger returns a random element in $\mathcal{X}$ if $b = 0$, and a random element in $\mathcal{L}$ if $b = 1$. At the end of the game, the attacker outputs a bit b' and wins the game if and only if $b' = b$.

Definition 4.1. A subset membership problem is said *hard* if, in the previous game, the advantage, which is defined as $\mathbf{Adv}^{\mathsf{memb}}_{q_m}(\mathcal{A}) := 2 \cdot \left| \Pr[b' = b] - 1/2 \right|$, is negligible for any PPT attacker $\mathcal{A}$.

Defining the subset membership hardness via the above game allows us to generically obtain tighter security bounds. In instantiations based on specific assumptions (e.g., DDH), the random self-reducibility of underlying problems allows avoiding any dependency on q_m in the reduction.

Smooth projective hash functions. Smooth projective hash functions (SPHFs) were introduced by Cramer and Shoup [2002] as a way to obtain chosen-ciphertext secure encryption schemes. We present below a variant tailored to fulfill our needs. A recent account on the different flavors of SPHFs can be found in [Benhamouda et al. 2013].

Definition 4.2. Using the previous notations, a *Smooth Projective Hash Function* (SPHF) is specified by a tuple of algorithms, $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$, of which the first one is probabilistic and the other algorithms are deterministic, defined as:

$\mathsf{HashKG}(1^\lambda)$. Let $\mathcal{K}$ denote the key space. On input of a security parameter λ, algorithm HashKG generates a hashing key $\mathsf{hk} \in \mathcal{K}$.
$\mathsf{ProjKG}(\mathsf{hk})$. Given a hashing key hk, this algorithm derives a projection key hp.
$\mathsf{Hash}(\mathsf{hk}, y)$. Given a hashing key hk and a word $y \in \mathcal{X}$, it outputs the hash value H of y.
$\mathsf{ProjHash}(\mathsf{hp}, y, w)$. Given a projection key hp, a word $y \in \mathcal{L}$ and a corresponding witness w (such that $\mathcal{R}(y, w) = 1$), it outputs the hash value H of y.

Further, letting Π denote the range of Hash and ProjHash and assuming that $(\Pi, \cdot)$ is an Abelian group (written multiplicatively with 1_Π as neutral element), an SPHF must satisfy the properties of *correctness* and of *special smoothness*:

Correctness. This property means that Hash and ProjHash hash to the same value for any word y in the language $\mathcal{L}$. More precisely, for every hashing key $\mathsf{hk} \xleftarrow{R} \mathsf{HashKG}(1^\lambda)$,

for all $y \in \mathcal{L}$ and associated witness w (such that $\mathcal{R}(y, w) = 1$), we have

$$\mathsf{Hash}(\mathsf{hk}, y) = \mathsf{ProjHash}(\mathsf{hp}, y, w)$$

provided $\mathsf{hp} = \mathsf{ProjHash}(\mathsf{hk})$.

Special smoothness. Let $\Pi' \subseteq \Pi$. Intuitively, the special smoothness says that the hash value of any $y \in \bar{\mathcal{L}} = \mathcal{X} \setminus \mathcal{L}$ looks random "over Π'", even knowing hp. Formally, an SPHF is said to be (ε_s, Π')-*smooth*, if for all $y \in \bar{\mathcal{L}}$, the following two distributions are ε_s-statistically indistinguishable:

$$D_0 = \left\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{R} \mathsf{HashKG}(1^\lambda),\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}),\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, y)\right\}$$

and

$$D_1 = \left\{(\mathsf{hp}, H \cdot H') \mid \mathsf{hk} \xleftarrow{R} \mathsf{HashKG}(1^\lambda),\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}),\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, y),\ H' \xleftarrow{R} \Pi'\right\} .$$

There are a couple of differences compared to the definition given in [Cramer and Shoup 2002]. Special smoothness replaces the original smoothness property [Cramer and Shoup 2002]. This latter property basically corresponds to the definition of special smoothness when $\Pi' = \Pi$. The special smoothness is also required to hold for *any* word $y \in \bar{\mathcal{L}}$, and not on average as in the original definition. Furthermore, only HashKG and Hash are required to be polynomial-time algorithms; ProjKG or $\mathsf{ProjHash}$ are not, in particular they may be unbounded.

An additional property that will be used in the security proofs is called *key uniformity*.

Key uniformity. An SPHF is said to have $\varepsilon_{\mathsf{hk}}$-*key-uniformity* when an honestly generated hashing key is $\varepsilon_{\mathsf{hk}}$-statistically indistinguishable from a random element in $\mathcal{K}$.

Key-homomorphic SPHF. An SPHF is said *key-homomorphic* if in addition

(1) $(\mathcal{K}, +)$ is an Abelian group written additively with $0_{\mathcal{K}}$ as neutral element;
(2) Π' is a subgroup of Π;
(3) for any two hashing keys $\mathsf{hk}_1, \mathsf{hk}_2 \in \mathcal{K}$ and for every word $y \in \mathcal{L}$:

$$\mathsf{Hash}(\mathsf{hk}_1 + \mathsf{hk}_2, y) = \mathsf{Hash}(\mathsf{hk}_1, y) \cdot \mathsf{Hash}(\mathsf{hk}_2, y) \ .$$

4.2. Our Abstract Scheme

We can now present our generic aggregator-oblivious encryption scheme, based on [specially] smooth, key-uniform, key-homomorphic SPHF.

Let f be an injective group homomorphism, $f : \mathbb{Z}_M \to \Pi'$. We assume that this homomorphism is efficiently invertible over the domain of possible sums X_t (which may be smaller than $\{0, \ldots, M-1\}$, as is the case for our DDH-based scheme).

$\mathsf{Setup}(1^\lambda)$. Let $\mathcal{L} \subset \mathcal{X}$ be a hard subset membership language. Let also a hash function $H : \mathbb{Z} \to \mathcal{X}$ (viewed as a random oracle in the security analysis). Finally, let n random hashing key $\mathsf{hk}_1, \ldots, \mathsf{hk}_n \xleftarrow{R} \mathcal{K}$. Define $\mathsf{hk}_0 = -\sum_{i=1}^n \mathsf{hk}_i$.

The system parameters are $\mathsf{param} = \{\mathcal{L}, H\}$ and the secret key of user i is $\mathsf{sk}_i = \mathsf{hk}_i$, with $0 \leq i \leq n$.

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$. At time period t, for a private input $x_{i,t} \in \mathbb{Z}_M$, user i produces

$$c_{i,t} = f(x_{i,t}) \cdot \mathsf{Hash}(\mathsf{hk}_i, H(t)) \in \Pi \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. The aggregator obtains the sum $X_t \pmod M$ for time period t by computing $X_t := f^{-1}\big(\mathsf{Hash}(\mathsf{hk}_0, H(t)) \cdot \prod_{i=1}^n c_{i,t}\big)$.

The correctness of the aggregation step follows from the homomorphic properties:

$$\begin{aligned}\mathsf{Hash}(\mathsf{hk}_0, H(t)) \cdot \prod_{i=1}^n c_{i,t} &= \prod_{i=0}^n \mathsf{Hash}(\mathsf{hk}_i, H(t)) \cdot \prod_{i=1}^n f(x_{i,t}) \\ &= \mathsf{Hash}(\sum_{i=0}^n \mathsf{hk}_i, H(t)) \cdot f(\sum_{i=1}^n x_{i,t}) \\ &= \mathsf{Hash}(0_{\mathcal{K}}, H(t)) \cdot f(\sum_{i=1}^n x_{i,t}) = 1_\Pi \cdot f(\sum_{i=1}^n x_{i,t}) \\ &= f(\sum_{i=1}^n x_{i,t})\end{aligned}$$

and therefore $X_t = \sum_{i=1}^n x_{i,t} \pmod M$ since f is injective.

4.3. Security

We prove security of the abstract scheme in the random oracle model, based on the hardness of the subset membership problem.

THEOREM 4.3. *The scheme provides* AO *security under the hard subset membership assumption of $\mathcal{L}$ in the random oracle model. Namely, for any probabilistic polynomial-time adversary $\mathcal{A}$, if the SPHF is (ε_s, Π')-smooth and $\varepsilon_{\mathsf{hk}}$-key-uniform, there exists a distinguisher $\mathcal{B}_{\mathsf{memb}}$ for the subset membership problem of $\mathcal{L}$ with comparable running time and such that*

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq 2e \cdot (q_{enc}+1) \cdot \left(\mathbf{Adv}_T^{\mathsf{memb}}(\mathcal{B}_{\mathsf{memb}}) + n\varepsilon_{\mathsf{hk}} + n\varepsilon_s + T\frac{|\mathcal{L}|}{|\mathcal{X}|} \right) ,$$

where n is the number of users, T is the number of periods queried to the random oracle, q_{enc} is the number of encryption queries made by the adversary for distinct periods other than $t^\star$ and e is the base for the natural logarithm.

We recall that we suppose $|\mathcal{L}|/|\mathcal{X}|$ is negligible.

PROOF. The proof proceeds with a sequence of several games. It begins with Game 0, which is the real game, and ends with Game 3, where even a computationally unbounded adversary has no advantage. For each $j \in \{0,1,2,3\}$, we denote by S_j the event that the challenger $\mathcal{B}$ outputs 1 in Game j. We also define $Adv_j = 2 \cdot |\Pr[S_j] - 1/2|$.

In the sequel, we assume w.l.o.g. that the adversary $\mathcal{A}$ has always already queried the random oracle H on input t before any encryption query for the time period t. We assume that the adversary does not query the random oracle H more than once for a given t.

Game 0. This is the real game. Namely, the challenger performs the setup of the system by randomly choosing $\mathsf{hk}_1, \ldots, \mathsf{hk}_n \xleftarrow{R} \mathcal{K}$ and defining $\mathsf{hk}_0 = -\sum_{i=1}^n \mathsf{hk}_i$. Queries to the random oracle H are answered by returning uniformly random group elements in $\mathcal{X}$. Encryption queries $(i,t,x_{i,t})$ for period t are answered by returning the ciphertext $c_{i,t} = f(x_{i,t}) \cdot H_{i,t}$ with $H_{i,t} = \mathsf{Hash}(\mathsf{hk}_i, H(t))$. Whenever the adversary decides to corrupt some player $i \in \{0,\ldots,n\}$, the challenger reveals hk_i. In the challenge phase, the adversary chooses a target time period $t^\star$, an uncorrupted subset $\mathcal{S}^\star \subseteq \mathcal{U}^\star$ and two distinct series $\langle (i, t^\star, x_{i,t^\star}^{(0)}) \rangle_{i \in \mathcal{S}^\star}$, $\langle (i, t^\star, x_{i,t^\star}^{(1)}) \rangle_{i \in \mathcal{S}^\star}$ which must satisfy Equation (1) (Section 2.1) if $\mathcal{S}^\star = \mathcal{U}^\star$ and the aggregator's private key sk_0 is exposed at some point of the game (see Section 2.1). At this stage, the challenger flips a fair binary coin $b \xleftarrow{R} \{0,1\}$ and the adversary $\mathcal{A}$ receives

$$\left\{ c_{i,t^\star} = f(x_{i,t^\star}^{(b)}) \cdot H_{i,t^\star} \right\}_{i \in \mathcal{S}^\star} \quad \text{with } H_{i,t^\star} = \mathsf{Hash}(\mathsf{hk}_i, H(t^\star)) \ .$$

We assume that the adversary queries $H(t^\star)$ before the challenge phase. Otherwise, $\mathcal{B}$ can simply make the query for itself. In the second phase, after a second series of queries, $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. We let the challenger $\mathcal{B}$ output 1 if $b' = b$ and 0 otherwise. The adversary's advantage in Game 0 is thus $Adv_0 = 2 \cdot |\Pr[S_0] - 1/2| = \mathbf{Adv}^{\mathsf{AO}}(\mathcal{A})$.

Game 1. This game is identical to Game 0 with the following difference. For each random oracle query $H(t)$, the challenger $\mathcal{B}$ flips a biased coin $\delta_t \in \{0,1\}$ that takes the value 1 with probability $1/(q_{enc}+1)$ and the value 0 with probability $q_{enc}/(q_{enc}+1)$. At the end of the game, $\mathcal{B}$ considers the event E that one of the following conditions holds:

— For the target time period $t^\star$, the coin $\delta_{t^\star}$ flipped for the hash query $H(t^\star)$ was $\delta_{t^\star} = 0$.
— There exists a time period $t \neq t^\star$ such that an encryption query $(i, t, .)$ was made for some user in $i \in \mathcal{U}^\star$ but for which $\delta_t = 1$.

If event E occurs (which $\mathcal{B}$ can detect at the end of the game), $\mathcal{B}$ halts and outputs a random bit. Otherwise, it outputs 1 if and only if $b' = b$. The same analysis as that of Coron [Coron 2000] shows that $\Pr[\neg E] = 1/e(q_{enc}+1)$, where e is the base for the natural logarithm. The transition from Game 0 to Game 1 is thus a transition based on a failure event of large probability [Dent 2006] and we thus have $Adv_1 = Adv_0 \cdot \Pr[\neg E] = Adv_0/e(q_{enc}+1)$.

Game 2. In this game, we modify the distribution of random oracle outputs. Specifically, the treatment of each hash query t depends on the random coin $\delta_t \in \{0,1\}$.

— If $\delta_t = 0$, the challenger $\mathcal{B}$ samples a random word $y_t \in \mathcal{L}$ together with a witness w_t, and defines $H(t) = y_t$;
— If $\delta_t = 1$, $\mathcal{B}$ samples a word $y_t \in \bar{\mathcal{L}}$ and defines $H(t) = y_t$.

It is straightforward to see that Game 2 and Game 1 are computationally indistinguishable if the hard subset membership assumption holds. Namely, there exists a distinguisher $\mathcal{B}_{\mathsf{memb}}$ for the subset membership problem of $\mathcal{L}$ with comparable running time and such that

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}_T^{\mathsf{memb}}(\mathcal{B}) + T\frac{|\mathcal{L}|}{|\mathcal{X}|}\,,$$

where the second term just comes from the fact that a random element in $\mathcal{X}$ is not in $\mathcal{L}$ with probability $1 - \frac{|\mathcal{L}|}{|\mathcal{X}|}$. Therefore $Adv_1 \leq Adv_2 + 2\big(\mathbf{Adv}_T^{\mathsf{memb}}(\mathcal{B}) + T\frac{|\mathcal{L}|}{|\mathcal{X}|}\big)$.

Game 3. In this game, $\mathcal{B}$ generates hk_i as $\mathsf{hk}_i \xleftarrow{R} \mathcal{K}$ (for $i \in \{1,\ldots,n\}$) instead of using HashKG ($\mathcal{B}$ is not polynomially bounded in this game), and:

— if $\sum_{i\in\mathcal{S}^\star} x_{i,t^\star}^{(0)} \neq \sum_{i\in\mathcal{S}^\star} x_{i,t^\star}^{(1)}$, then we set $H_{i,t^\star} = \mathsf{Hash}(\mathsf{hk}_i, H(t^\star)) \cdot H_i'$ with $H_i' \xleftarrow{R} \Pi'$, for $i \in \mathcal{S}^\star$; in this case, clearly, H_i' completely masks $f(x_{i,t^\star}^{(b)})$;
— otherwise, let $i^\star = \max \mathcal{S}^\star$, then we set $H_{i,t^\star} = \mathsf{Hash}(\mathsf{hk}_i, H(t^\star)) \cdot H_i'$ with $H_i' \xleftarrow{R} \Pi'$ for $i \in \mathcal{S}^\star \setminus \{i^\star\}$, and $H_{i^\star,t^\star} = \prod_{i\in\{0,\ldots,n\}\setminus\{i^\star\}} H_{i,t^\star}^{-1}$ (which is equal to $\mathsf{Hash}(\mathsf{hk}_{i^\star}, H(t^\star)) \cdot H_{i^\star}'$, with $H_{i^\star}' = \prod_{i\in\mathcal{S}^\star\setminus\{i^\star\}} H_i'^{-1}$); we have that, for any sequence $(H_i')_{i\in\mathcal{S}^\star}$, there exists a unique $(H_i'')_{i\in\mathcal{S}^\star}$ (with $\prod_{i\in\mathcal{S}^\star} H_i' = \prod_{i\in\mathcal{S}^\star} H_i'' = 1$), that satisfies $H_i' \cdot f(x_{i,t^\star}^{(0)}) = H_i'' \cdot f(x_{i,t^\star}^{(1)})$.

It is therefore clear that $\Pr[S_3] = 1/2$, so that $\mathcal{A}$ has no advantage.
In addition, in Lemma B.1 (Appendix B.1), we show that

$$|\Pr[S_3] - \Pr[S_2]| \leq n\varepsilon_{\mathsf{hk}} + n\varepsilon_s\,,$$

and so $Adv_3 \leq Adv_2 + 2n\varepsilon_{\mathsf{hk}} + 2n\varepsilon_s$.

Putting all together, we get

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq 2e \cdot (q_{enc}+1) \cdot \Big(\mathbf{Adv}^{\mathsf{memb}}(\mathcal{B}) + n\varepsilon_{\mathsf{hk}} + n\varepsilon_s + T\frac{|\mathcal{L}|}{|\mathcal{X}|}\Big)\,. \quad \square$$

4.4. A concrete instantiation

An example of hard subset membership language is the DDH language. We then have:

$$\mathcal{X} = \mathbb{G} \times \mathbb{G}\,, \quad \mathcal{L} = \left\{\vec{y} = (y_1, y_2) \in \mathcal{X} \mid \exists w \in \mathbb{Z}_p \text{ such that } \mathcal{R}(\vec{y}, w) = 1\right\}$$

with

$$\mathcal{R}(\vec{y}, w) = 1 \iff y_1 = g^w \text{ and } y_2 = h^w\,,$$

where g and h are two generators of a group $\mathbb{G}$ of prime order p. The hard subset membership assumption for this language is the DDH assumption.

For the DDH language, Cramer and Shoup construct an SPHF as follows. The key space is $\mathcal{K} = \mathbb{Z}_p \times \mathbb{Z}_p$ and the hash range is $\Pi = \mathbb{G}$. The hashing key is a random tuple $\mathsf{hk} = (s, t) \in \mathcal{K}$ and the projection key is $\mathsf{hp} = g^s h^t$. We then have:

$$\mathsf{Hash}\colon \mathcal{K} \times \mathcal{X} \to \Pi,\ \big(\mathsf{hk}, (y_1, y_2)\big) \mapsto \mathsf{Hash}\big(\mathsf{hk}, (y_1, y_2)\big) = {y_1}^s {y_2}^t$$

and, if w is a witness for (y_1, y_2), i.e., $y_1 = g^w$ and $y_2 = h^w$,

$$\mathsf{ProjHash}\colon \mathbb{G} \times \mathcal{X} \times \mathbb{Z}_p \to \Pi,\ \big(\mathsf{hp}, (y_1, y_2), w\big) \mapsto \mathsf{ProjHash}\big(\mathsf{hp}, (y_1, y_2), w\big) = \mathsf{hp}^w\ .$$

This SPHF is $(0, \Pi)$-smooth (see [Cramer and Shoup 2002]). It is readily seen that this SPHF is key-homomorphic and 0-key-uniform. If we set $f : \mathbb{Z}_p \to \Pi, x \mapsto f(x) = g^x$, which clearly is an injective group homomorphism, we get exactly our DDH-based scheme of Section 3. Observe also that plugging $\varepsilon_{\mathsf{hk}} = \varepsilon_s = 0$, $|\mathcal{X}| = p^2$, and $|\mathcal{L}| = p$ in Theorem 4.3 yields the statement of Theorem 3.2.

5. FURTHER INSTANTIATIONS

5.1. k-linear assumption and generalizations

In [2013], Escala *et al.* generalize the k-LIN assumptions [Boneh et al. 2004; Hofheinz and Kiltz 2007; Shacham 2007] in an assumption called MDDH. They also show how to construct an SPHF from any MDDH assumption. Their construction can be used directly in our framework, with $f(x) = g^x$ (which is invertible for x in small ranges), since their SPHFs are clearly key-homomorphic and 0-key-uniform. This yields in particular an aggregator oblivious encryption scheme from the k-LIN assumption.

When $k = 1$, since 1-LIN = DDH, we get exactly the new scheme presented in Section 3. A larger value of k implies a weaker assumption. The drawback is an increase of the private-key size, which has to be multiplied by $(k+1)/2$. The other parameter sizes given in Table 1 are unchanged.

5.2. SD assumption

Yet another instantiation can be derived from the subgroup decision (SD) assumption (for cyclic groups of composite order, without pairing) as introduced by Boneh et al. [2005].

Let $N = pq$ be an RSA modulus with p and q two large primes. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order N, $\mathbb{G}_p = \langle g_p \rangle$ be the subgroup of order p, and $\mathbb{G}_q = \langle g_q \rangle$ be the subgroup of order q. Define $\mathcal{X} = \mathbb{G}$, $\mathcal{L} = \mathbb{G}_p$, $\mathcal{K} = \mathbb{Z}_N$, $\Pi = \mathbb{G}$, and $\Pi' = \mathbb{G}_q$. A witness w for $y \in \mathcal{L}$ is a discrete logarithm of y in base g_p, $y = {g_p}^w$. Then set:

— for any $y \in \mathcal{X}$, $\mathsf{Hash}(\mathsf{hk}, y) = y^r$, and for any $y \in \mathcal{L}$, $\mathsf{ProjHash}(\mathsf{hp}, y, w) = \mathsf{hp}^w$ with w a witness for y such that $y = {g_p}^w$, for $\mathsf{hk} = r \xleftarrow{R} \mathcal{K}$ and $\mathsf{hp} = {g_p}^{\mathsf{hk}}$;
— $f(x) = {g_q}^x$ for $x \in \mathbb{Z}_q$.

The so-obtained SPHF is $(0, \Pi')$-smooth, 0-key-uniform, and key-homomorphic.

5.3. DCR assumption

Paillier's decision composite residuosity (DCR) assumption [Paillier 1999] can also be used to instantiate a slight variant of our general framework. The resulting aggregator-oblivious

encryption scheme shares many similarities with the Joye-Libert's construction in [Joye and Libert 2013] but it is not strictly the same scheme.

We rely on a variant of the hash proof system proposed by Cramer and Shoup [2002],with inefficient ProjKG and $\mathsf{ProjHash}$. Let $N = pq$ be an RSA modulus where p and q are two distinct primes. Define

$$\mathcal{X} = (\mathbb{Z}_{N^2})^* , \quad \mathcal{L} = \{y = z^N \mid z \in \mathcal{X}\} \subset \mathcal{X} , \quad \mathcal{K} = \mathbb{Z}_{N\phi(N)} ,$$
$$\Pi = \mathcal{X} , \quad \Pi' = \langle 1 + N \rangle \subset \Pi ,$$

and set

— For any $y \in \mathcal{X}$, $\mathsf{Hash}(\mathsf{hk}, y) = y^r$, and for any $y \in \mathcal{L}$, $\mathsf{ProjHash}(\mathsf{hp}, y, \perp) = y^{\mathsf{hp}}$, where the hashing key is chosen as $\mathsf{hk} = r \xleftarrow{R} \{-2^\kappa N^2, \ldots, 2^\kappa N^2\}$ (with κ depending on the expected security —see below) and $\mathsf{hp} = r \bmod \phi(N)$ —since $\phi(N)$ is unknown (otherwise the language is not hard subset membership), note that hp cannot be efficiently computed;
— $f(x) = (1 + N)^x \in \Pi'$ for $x \in \mathbb{Z}_N$.

The resulting SPHF is clearly key-homomorphic. Further, $r \bmod N\phi(N)$ is $1/(2^{\lambda+1}n)$-statistically close from the uniform distribution over $\mathcal{K}$. Since if $r \xleftarrow{R} \mathcal{K}$, the SPHF would be $(0, \Pi')$-smooth, the actual resulting SPHF is $(1/(2^{\lambda+1}n), \Pi')$-smooth. As shown in Appendix B.2, the corresponding scheme provably meets the notion of aggregator obliviousness. We have the following theorem:

THEOREM 5.1. *The scheme provides* AO *security under the* DCR *assumption in the random oracle model. Specifically, for any probabilistic polynomial-time adversary* $\mathcal{A}$*, there exists a* DCR *distinguisher* $\mathcal{B}_{\mathsf{DCR}}$ *with comparable running time and such that*

$$\mathbf{Adv}^{\mathsf{AO}}(\mathcal{A}) \leq 2e \cdot (q_{enc} + 1) \cdot \left(\mathbf{Adv}^{DCR}(\mathcal{B}) + \frac{n}{2^\kappa} + \frac{T}{\phi(N)} \right) ,$$

where q_{enc} *is the number of encryption queries made by the adversary for distinct periods other than* $t^\star$*,* T *is the maximum number of periods and* e *is the base for the natural logarithm.* □

REFERENCES

Gergely Ács and Claude Castelluccia. 2011. I Have a DREAM! (DiffeRentially privatE smArt Metering). In *Information Hiding (IH 2011) (LNCS)*, Tomás Filler et al. (Eds.), Vol. 6958. Springer, 118–132.

Mihir Bellare and Phillip Rogaway. 1996. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EUROCRYPT'96 (LNCS)*, Ueli M. Maurer (Ed.), Vol. 1070. Springer, 399–416.

Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. 2013. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In *CRYPTO 2013, Part I (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8042. Springer, 449–475. DOI:http://dx.doi.org/10.1007/978-3-642-40041-4_25

Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *CRYPTO 2004 (LNCS)*, Matthew Franklin (Ed.), Vol. 3152. Springer, 41–55.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-DNF Formulas on Ciphertexts. In *TCC 2005 (LNCS)*, Joe Kilian (Ed.), Vol. 3378. Springer, 325–341.

T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2012. Privacy-Preserving Stream Aggregation with Fault Tolerance. In *FC 2012 (LNCS)*, Angelos D. Keromytis (Ed.), Vol. 7397. Springer, 200–214.

Jean-Sébastien Coron. 2000. On the Exact Security of Full Domain Hash. In *CRYPTO 2000 (LNCS)*, Mihir Bellare (Ed.), Vol. 1880. Springer, 229–235.

Jean-Sébastien Coron. 2002. Optimal Security Proofs for PSS and Other Signature Schemes. In *EUROCRYPT 2002 (LNCS)*, Lars R. Knudsen (Ed.), Vol. 2332. Springer, 272–287.

Ronald Cramer and Victor Shoup. 2002. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *EUROCRYPT 2002 (LNCS)*, Lars R. Knudsen (Ed.), Vol. 2332. Springer, 45–64.

Alexander W. Dent. 2006. A Note On Game-Hopping Proofs. Cryptology ePrint Archive, Report 2006/260. (2006). http://eprint.iacr.org/.

Cynthia Dwork. 2008. Differential privacy: A survey of results. In *Theory and applications of models of computation*. Springer, 1–19.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT 2006 (LNCS)*, Serge Vaudenay (Ed.), Vol. 4004. Springer, 486–503.

ECRYPT II. 2012. Yearly Report on Algorithms and Keysizes. (2012).

Zekeriya Erkin and Gene Tsudik. 2012. Private Computation of Spatial and Temporal Power Consumption with Smart Meters. In *ACNS 12 (LNCS)*, Feng Bao, Pierangela Samarati, and Jianying Zhou (Eds.), Vol. 7341. Springer, 561–577.

Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. 2013. An Algebraic Framework for Diffie-Hellman Assumptions. In *CRYPTO 2013, Part II (LNCS)*, Ran Canetti and Juan A. Garay (Eds.), Vol. 8043. Springer, 129–147. DOI:http://dx.doi.org/10.1007/978-3-642-40084-1_8

Flavio D. Garcia and Bart Jacobs. 2010. Privacy-Friendly Energy-Metering via Homomorphic Encryption. In *Security and Trust Management (STM 2010) (LNCS)*, Jorge Cuéllar et al. (Eds.), Vol. 6710. Springer, 226–238.

Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. 2013. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th FOCS*. IEEE Computer Society Press, 40–49.

Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. 2014. Multi-input Functional Encryption. In *EUROCRYPT 2014 (LNCS)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.), Vol. 8441. Springer, 578–602. DOI:http://dx.doi.org/10.1007/978-3-642-55220-5_32

Dennis Hofheinz and Eike Kiltz. 2007. Secure Hybrid Encryption from Weakened Key Encapsulation. In *CRYPTO 2007 (LNCS)*, Alfred Menezes (Ed.), Vol. 4622. Springer, 553–571.

Marek Jawurek and Florian Kerschbaum. 2012. Fault-tolerant privacy-preserving statistics. In *Privacy Enhancing Technologies (PETS 2012) (LNCS)*, Simone Fischer-Hübner and Matthew Wright (Eds.), Vol. 7384. Springer, 221–238.

Marek Jawurek, Florian Kerschbaum, and George Danezis. 2012. *Privacy Technologies for Smart Grids – A Survey of Options*. Technical Report MSR-TR-2012-119. Microsoft Research.

Marc Joye and Benoît Libert. 2013. A Scalable Scheme for Privacy-Preserving Aggregation of Time-Series Data. In *FC 2013 (LNCS)*, Ahmad-Reza Sadeghi (Ed.), Vol. 7859. Springer, 111–125. DOI:http://dx.doi.org/10.1007/978-3-642-39884-1_10

Klaus Kursawe, George Danezis, and Markulf Kohlweiss. 2011. Privacy-Friendly Aggregation for the Smart-Grid. In *Privacy Enhancing Technologies (PETS 2011) (LNCS)*, Simone Fischer-Hübner and Nicholas Hopper (Eds.), Vol. 6794. Springer, 221–238.

Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. 2014. Private and Dynamic Time-Series Data Aggregation with Trust Relaxation. In *CANS 14 (LNCS)*, Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis (Eds.), Vol. 8813. Springer, 305–320. DOI:http://dx.doi.org/10.1007/978-3-319-12280-9_20

Patrick McDaniel and Stephen McLaughlin. 2009. Security and Privacy Challenges in the Smart Grid. *IEEE Security & Privacy* 7, 3 (May/June 2009), 75–77.

Ravi Montenegro and Prasad Tetali. 2009. How long does it take to catch a wild kangaroo?. In *41st ACM STOC*, Michael Mitzenmacher (Ed.). ACM Press, 553–560.

Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT'99 (LNCS)*, Jacques Stern (Ed.), Vol. 1592. Springer, 223–238.

Vibhor Rastogi and Suman Nath. 2010. Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption. In *2010 ACM SIGMOD International Conference on Management of Data (SIGMOD 2010)*, Ahmed K. Elmagarmid and Divyakant Agrawal (Eds.). ACM Press, 735–746.

Hovav Shacham. 2007. A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants. Cryptology ePrint Archive, Report 2007/074. (2007). http://eprint.iacr.org/.

Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. 2011. Privacy-Preserving Aggregation of Time-Series Data. In *NDSS 2011*. The Internet Society.

A. AO ENCRYPTION SCHEMES

For completeness, we review in this appendix the two known constructions for aggregator-oblivious encryption [Shi et al. 2011; Joye and Libert 2013].

A.1. Shi *et al.*'s scheme

$\mathsf{Setup}(1^\lambda)$. Let $\mathbb{G}$ be a group of prime order p for which the DDH assumption holds, and let a random generator $g \in \mathbb{G}$. Let also a hash function $H : \mathbb{Z} \to \mathbb{G}$ viewed as a random oracle. Finally, let n random elements in $\mathbb{Z}_p$, $s_1, \ldots, s_n$, and define $s_0 = -\sum_{i=1}^n s_i \bmod p$.

The system parameters are $\mathsf{param} = \{\mathbb{G}, g, H\}$ and the secret key of user i, $0 \leq i \leq n$, is $\mathsf{sk}_i = s_i$.

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$. At time period t, for a private input $x_{i,t} \in \mathbb{Z}_p$, user i produces

$$c_{i,t} = g^{x_{i,t}} H(t)^{s_i} \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. The aggregator obtains the sum X_t for time period t by first computing $V_t := H(t)^{s_0} \prod_{i=1}^n c_{i,t} = g^{X_t}$ and next the discrete logarithm of V_t w.r.t. basis g.

[Notice that since g has order p, the so-obtained value for X_t is defined modulo $M = p$.]

A.2. Joye-Libert's scheme

$\mathsf{Setup}(1^\lambda)$. Let $M = N = pq$ be an RSA modulus of length ℓ, i.e., a product of two random equal-size primes p, q. Note that the size condition on p and q implies that $\gcd(\phi(N), N) = 1$. Let also a hash function $H : \mathbb{Z} \to \mathbb{Z}_{N^2}^*$ viewed as a random oracle. Finally let n randomly chosen elements in $\pm\{0,1\}^{2\ell}$, $s_1, \ldots, s_n$, and define $s_0 = -\sum_{i=1}^n s_i$.

The system parameters are $\mathsf{param} = \{N, H\}$ and the secret key of user i, $0 \leq i \leq n$, is $\mathsf{sk}_i = s_i$.

$\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, t, x_{i,t})$. At time period t, for a private input $x_{i,t} \in \mathbb{Z}_N$, user i produces

$$c_{i,t} = (1 + x_{i,t}N) \cdot H(t)^{s_i} \bmod N^2 \ .$$

$\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, t, c_{1,t}, \ldots, c_{n,t})$. The aggregator obtains the sum X_t for time period t by first computing $V_t := H(t)^{s_0} \prod_{i=1}^n c_{i,t} \bmod N^2$ and next X_t as $X_t = \frac{V_t - 1}{N}$.

[Notice that since $(1 + N)$ has order N and $(1 + x_{i,t}N) \equiv (1 + N)^{x_{i,t}} \pmod{N^2}$, the so-obtained value for X_t is defined modulo $M = N$.]

B. DEFERRED PROOFS

B.1. Indistinguishability of Games 2 and 3

LEMMA B.1. *If the SPHF is (ε_s, Π')-smooth, $\varepsilon_{\mathsf{hk}}$-key-uniform, and key-homomorphic, Game 2 and Game 3 are $(n\varepsilon_s + n\varepsilon_{\mathsf{hk}})$-statistically-indistinguishable.*

PROOF. We write $\mathcal{S}^\star = \{i_1, \ldots, i_m\}$.

We first suppose that, in Game 2, all hashing keys hk_i are chosen uniformly at random in $\mathcal{K}$ instead of being generated using HashKG, as in Game 3. Thanks to the $\varepsilon_{\mathsf{hk}}$-key-uniformity, this is $n\varepsilon_{\mathsf{hk}}$-statistically indistinguishable.

The main idea of the proof consists in remarking that the hashing keys hk_{i_j} are not needed to compute $H_{i_j,t}$ for $t \neq t^\star$, but only the projection keys hp_{i_j}, since $H_{i_j,t} = \mathsf{ProjHash}(\mathsf{hp}_{i_j}, H(t), w_t)$.

Consider first the case $\sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(0)} = \sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(1)}$. We suppose $i^\star = i_m$. We remark that $H_{i_m,t}$ (for t such that $\delta_t = 0$) can even be computed without knowing hp_{i_m}, simply by computing it as $H_{i_m,t} = \prod_{i \in \{0,\ldots,n\} \setminus \{i_m\}} H_{i,t}^{-1}$. Therefore, even if hk_0 is public, no

information is leaked about hk_{i_m}, and the only information leaked about $\mathsf{hk}_{i_1}, \ldots, \mathsf{hk}_{i_{m-1}}$ is $\mathsf{hp}_{i_1}, \ldots, \mathsf{hp}_{i_{m-1}}$, in Game 2 (in the challenge phase), as the only other relation verified by hk_{i_j} is $\sum_{i=0}^{n} \mathsf{hk}_i = 0_{\mathcal{K}}$. Special smoothness on $\mathsf{hk}_{i_1}, \ldots, \mathsf{hk}_{i_{m-1}}$ so ensures that $\mathsf{Hash}(\mathsf{hk}_{i_j}, H(t^\star))$ is indistinguishable from $\mathsf{Hash}(\mathsf{hk}_{i_j}, H(t^\star)) \cdot H'_{i_j}$, with $H'_{i_j} \xleftarrow{R} \Pi'$ and $j \in \{1, \ldots, m-1\}$. Hence Game 2 and Game 3 are $(m-1)\varepsilon_s$-statistically indistinguishable, and so $n\varepsilon_s$-statistically indistinguishable, as $m \leq n+1$ in that case.

Suppose now that $\sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(0)} \neq \sum_{i \in \mathcal{S}^\star} x_{i,t^\star}^{(1)}$. Then either the aggregator is not corrupted (and we set $i^\dagger = 0$), or there exists some uncorrupted user $i^\dagger \in \mathcal{U}^\star \setminus \mathcal{S}^\star$. We then remark as above that $H_{i^\dagger,t}$ (for t such that $\delta_t = 0$) can even be computed without knowing $\mathsf{hp}_{i^\dagger}$, simply by computing it as $H_{i^\dagger,t} = \prod_{i \in \{0,\ldots,n\} \setminus \{i^\dagger\}} H_{i,t}^{-1}$. Therefore, no information is leaked about $\mathsf{hk}_{i^\dagger}$, and the only information leaked about $\mathsf{hk}_{i_1}, \ldots, \mathsf{hk}_{i_m}$ is $\mathsf{hp}_{i_1}, \ldots, \mathsf{hp}_{i_m}$, in Game 2 (in the challenge phase). Thus, special smoothness on $\mathsf{hk}_{i_1}, \ldots, \mathsf{hk}_{i_m}$ ensures that $\mathsf{Hash}(\mathsf{hk}_{i_j}, H(t^\star))$ is indistinguishable from $\mathsf{Hash}(\mathsf{hk}_{i_j}, H(t^\star)) \cdot H'_{i_j}$, with $H'_{i_j} \xleftarrow{R} \Pi'$ and $j \in \{1, \ldots, m\}$. Hence Game 2 and Game 3 are $m\varepsilon_s$-statistically indistinguishable, and so $n\varepsilon_s$-statistically indistinguishable, as $m \leq n$ in that case.

Finally, Game 2 (original version, with hk_i generated with HashKG) and Game 3 are $(n\varepsilon_s + n\varepsilon_{\mathsf{hk}})$-statistically indistinguishable. □

B.2. Proof of our DCR Instantiation

Proof Theorem 5.1. The proof of our DCR instantiation in Section 5.3 is similar to the proof of Theorem 4.3, except for Lemma B.1 which is no more applicable, since the scheme is not key-uniform (as hashing keys have multiple representations), and hk_0 is the sum of the hk_i over the integers and not in $\mathcal{K}$.

We will show that in the proof of Theorem 4.3, Game 2 and Game 3 are $\frac{n}{2^\kappa}$-statistically indistinguishable. This will conclude the proof, as $|\mathcal{L}|/|\mathcal{X}| = 1/\phi(N)$.

First, when the aggregator is not corrupted, we show that the proof of Lemma B.1 still works. In this case, hk_0 is not known, and we can consider two games Game 2′ and Game 3′ similar to Game 2 and Game 3, except that the hashing keys hk_i for $i \in \mathcal{U}^\star$ are drawn uniformly from $\mathcal{K}$, instead of from $\{-2^\kappa N^2, \ldots, 2^\kappa N^2\}$. Game 2′ and Game 3′ are $n/2^{\kappa+1}$-statistically indistinguishable from Game 2 and Game 3 respectively (for a fixed set $\mathcal{U}^\star$, when the aggregator is not corrupted), as the adversary never sees hk_i but only $\mathsf{hk}_i \bmod N\phi(N)$, and when $\mathsf{hk}_i \xleftarrow{R} \{-2^\kappa N^2, \ldots, 2^\kappa N^2\}$, $\mathsf{hk}_i \bmod N\phi(N)$ is $n/2^{\kappa+1}$-statistically close to $\mathsf{hk}_i \xleftarrow{R} \mathcal{K}$. And we can show as in Lemma B.1, that Game 2′ and Game 3′ are identical since the SPHF is $(0, \Pi')$-smooth when its hashing keys are drawn uniformly from $\mathcal{K}$. So finally, in that case, Game 2 and Game 3 are $n/2^\kappa$-statically indistinguishable.

Second, we suppose that the aggregator is corrupted, and so hk_0 is known to the adversary. Necessarily, $\sum_{i \in \mathcal{S}^\star} x_{i,t}^{(0)} \neq \sum_{i \in \mathcal{S}^\star}^{(1)} x_{i,t}^{(1)}$ and there exists some uncorrupted user $i^\dagger \in \mathcal{U}^\star \setminus \mathcal{S}^\star$. To have a proof along he lines of Lemma B.1, we just need to show that $\mathsf{hk}_{i_1}, \ldots, \mathsf{hk}_{i_m}$ could be drawn uniformly from $\mathcal{K}$ instead of from $\{-2^\kappa N^2, \ldots, 2^\kappa N^2\}$, as above. That would conclude the proof.

More precisely, suppose that $i_1 = 1, \ldots, i_m = m$ and $i^\dagger = n$, for the sake of simplicity, and define the following probability distributions (for $j = 0, \ldots, m$):

$$\mathcal{D}_j = \Big\{ (\mathsf{hk}_0, \mathsf{hk}'_1, \ldots, \mathsf{hk}'_m, \mathsf{hk}_{m+1}, \ldots, \mathsf{hk}_{n-1}) \mid \mathsf{hk}_1, \ldots, \mathsf{hk}_n \xleftarrow{R} \{-2^\kappa N^2, \ldots, 2^\kappa N^2\}, \\ \mathsf{hk}_0 \leftarrow \sum_{i=1}^{n} \mathsf{hk}_i, \mathsf{hk}'_1, \ldots, \mathsf{hk}'_j \xleftarrow{R} \mathcal{K},\ \mathsf{hk}'_{j+1} \leftarrow \mathsf{hk}_{j+1} \bmod N\Phi(N), \\ \ldots, \mathsf{hk}'_m \leftarrow \mathsf{hk}_m \bmod N\Phi(N) \Big\} .$$

We just need to prove that $\mathcal{D}_j$ is $1/2^{\kappa+1}$-close to $\mathcal{D}_{j+1}$, hence $\mathcal{D}_0$ is $n/2^{\kappa+1}$ close to $\mathcal{D}_m$ ($m \leq n$). Since $\mathcal{D}_0$ corresponds to the way the hashing keys are generated in Game 2 and Game 3, and in $\mathcal{D}_n$, $\mathsf{hk}'_1, \ldots, \mathsf{hk}'_m$ (corresponding to the hashing keys for the non-corrupted users modulo $N\phi(N)$) are totally independent of hk_0 and the other hk_i's, the proof of Lemma B.1 can be used (with an additive term of $2n/2^{\kappa+1}$).

To prove that $\mathcal{D}_j$ is $1/2^{\kappa+1}$-close to $\mathcal{D}_{j+1}$, we need the following lemma:

LEMMA B.2. *Let M, N' be two integers, with $M \geq N'$. Let X, Y be two random uniform random variables in $\{-M, \ldots, M\}$, and X' be a uniform random variable in $\{0, \ldots, N'-1\}$. Let us suppose that X, Y, X' are mutually independent. Then the statistical distance between the distribution of $(X \bmod N',\ X+Y)$ and of $(X',\ X+Y)$ is at most $N'/(2(2M+1))$.*

PROOF. This statistical distance is

$$\frac{1}{2} \sum_{\substack{x' \in \{0,\ldots,N'\} \\ z \in \{-2M,\ldots,2M\}}} \left| \Pr[X \bmod N' = x',\ X+Y=z] - \Pr[X'=x',\ X+Y=z] \right| \ .$$

We have X', X, Y are mutually independent, so

$$\Pr[X'=x',\ X+Y=z] = \Pr[X'=x'] \cdot \Pr[X+Y=z] = \frac{1}{N'} \cdot \frac{2M+1-|z|}{(2M+1)^2} \ .$$

Moreover,

$$\begin{aligned} \Pr[X \bmod N' = x',\ X+Y=z] &= \sum_{\substack{x \in \{-M,\ldots,M\} \\ \text{such that } x \bmod N' = x'}} \Pr[X=x,\ X+Y=z] \\ &= \sum_{\substack{x \in \{-M,\ldots,M\} \\ \text{such that } x \bmod N' = x'}} \Pr[X=x,\ Y=z-x] \\ &= \frac{n}{(2M+1)^2} \end{aligned}$$

where n is the number of values $x \in \{-M, \ldots, M\}$ such that $x \bmod N' = x'$ and $z - x \in \{-M, \ldots, M\}$. There are $2M+1-|z|$ values x such that $z - x \in \{-M, \ldots, M\}$, namely $-M, \ldots, M+z$ if $z \leq 0$, and $-M+z, \ldots, M$ if $z \geq 0$. Among these values either $\lfloor (2M+1-|z|)/N' \rfloor$ or $\lfloor (2M+1-|z|)/N' \rfloor + 1$ of them are such that $x \bmod N' = x'$. Therefore, the statistical distance satisfies

$$\frac{1}{2} \sum_{x',z} \left| \frac{n}{(2M+1)^2} - \frac{2M+1-|z|}{N'(2M+1)^2} \right| \leq \frac{1}{2} \sum_{x',z} \frac{1}{(2M+1)^2} = \frac{N'}{2(2M+1)} \ .$$

□

Back to the theorem, we suppose hk_i known by the adversary and we fix them, except for $i \in \{0, j, j+1\}$. We then show that the conditional distributions $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ given these values, are $n/2^{\kappa+1}$-close, which is stronger than showing than the original distributions are $1/2^{\kappa+2}$-close, hence $1/2^{\kappa+1}$-close. We do that by applying the previous lemma with $X = \mathsf{hk}_j$, $Y = \mathsf{hk}_{j+1}$, $\mathsf{hk}_0 = \sum_{i=1}^{n} \mathsf{hk}_i = \mathsf{hk}_1 + \cdots + \mathsf{hk}_{j-1} + X + Y + \mathsf{hk}_{j+2} + \cdots + \mathsf{hk}_n$ (each term of this sum is constant except X and Y), $N' = N\phi(N)$, and $M = 2^\kappa N^2 \geq 2^\kappa N'$. □

C. IMPOSSIBILITY RESULT OF TIGHTNESS FOR A PREVIOUS SCHEME

We now show that any blackbox non-rewinding reduction from the aggregator obliviousness of Shi *et al.*'s scheme in [2011] to a non-interactive problem loses a factor of at least n^2. For

that purpose, we will outline a meta-reduction as in [Coron 2002]. The idea is to show that any reduction losing a factor better than (about) n^2 can be converted into an adversary for the original hard problem, by constructing an adversary $\mathcal{B}$ which acts as an adversary for the reduction but which can also rewind the reduction.

The basic idea is that, in the scheme of Shi *et al.*, sk_i is completely defined (from the information theory viewpoint) when a ciphertext $c_{i,1}$ for 0 (for example) is given; and sk_0 is defined by $(c_{i,1})_i$.

More precisely, suppose that the reduction can solve the original hard problem with probability ε_R, when playing with any adversary breaking the aggregator obliviousness with advantage $2\varepsilon_{\mathcal{A}} - 1$. Let us construct an adversary $\mathcal{B}$ (which has the right to rewind the reduction) as follows: $\mathcal{B}$ will first ask for the aggregator secret key sk_0 and for ciphertexts $c_{i,1}$ of 0 for time period 1 (and for each user i). It can check that any secret key sk_i given by the reduction is valid or not (for $i \notin \{i_1, i_2\}$), with respect to $c_{i,1}$ (by checking that $\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, 1, 0) = c_{i,t}$) and that sk_0 is valid (by checking that $\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, 1, c_{1,1}, \ldots, c_{n,1}) = 0$). Then it will choose two random users $i_1 \neq i_2$. For all pair of distinct users $\{i_3, i_4\} \neq \{i_1, i_2\}$, it then asks all the secret keys sk_i (in an arbitrary order) except i_3 and i_4, store them, and then rewind the adversary just after the corruption of sk_0. After that, $\mathcal{B}$ will have stored $(n-1)(n-2)/2$ keys sk_{i_1} and $(n-1)(n-2)/2$ keys sk_{i_2}. If none of them are valid, $\mathcal{B}$ aborts and returns $b' = 0$ with probability $1/2$, and $b' = 1$ otherwise.

Otherwise, $\mathcal{B}$ then asks all the secret keys sk_i except i_1 and i_2. If none of them are valid, $\mathcal{B}$ aborts and returns $b' = 0$ with probability $1/2$, and $b' = 1$ otherwise. Otherwise, it just submits the challenge: $\mathcal{S}^\star = \{i_1, i_2\}$, $t^\star = 2$, $(x^{(0)}_{i_1,2} = 0, x^{(0)}_{i_2,2} = 1)$ and $(x^{(1)}_{i_1,2} = 1, x^{(1)}_{i_2,2} = 0)$. The reduction will returns a pair of ciphertexts $\langle c_{i_1,1}, c_{i_1,2}\rangle$ encrypting either $\langle x^{(0)}_{i_1,2}, x^{(0)}_{i_2,2}\rangle$ or $\langle x^{(1)}_{i_1,2}, x^{(1)}_{i_2,2}\rangle$. And $\mathcal{B}$ will check that these ciphertexts are coherent with sk_0, by computing $c_{i,2} = \mathsf{Enc}(\mathsf{param}, \mathsf{sk}_i, 2, 0)$ for all $i \notin \{i_1, i_2\}$, and checking that $\mathsf{AggrDec}(\mathsf{param}, \mathsf{sk}_0, 2, c_{1,2}, \ldots, c_{n,2}) = 1$. Finally, if $\mathcal{B}$ got a valid secret key sk_{i_1} and if $\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_{i_1}, 2, 0) = c_{i_1,2}$, $\mathcal{B}$ sets $b'' = 0$; if $\mathcal{B}$ got a valid secret key sk_{i_2} and $\mathsf{Enc}(\mathsf{param}, \mathsf{sk}_{i_2}, 2, 1) = i_2, 1$, $\mathcal{B}$ sets $b'' = 0$; otherwise, $\mathcal{B}$ sets $b'' = 1$. And $\mathcal{B}$ outputs $b' = b''$ with probability $\varepsilon_{\mathcal{A}}$ and $b' = 1 - b''$ otherwise.

We remark that if sk_{i_1} or sk_{i_2} is valid, $\mathcal{B}$ would behave exactly as an all powerful (non polynomially bounded) adversary $\mathcal{A}$ which would do the same as $\mathcal{B}$, except it does not perform any rewinding (and so never corrupts sk_{i_1} and sk_{i_2}), and instead computes sk_{i_1} and sk_{i_2} simply by trying all possible values. This is true thanks to the check with sk_0 which ensures that if $c_{i_1,2}$ is an encryption of 0 (respectively 1), then $c_{i_2,2}$ is an encryption of 1 (respectively 0), and so knowing only one of sk_{i_1} and sk_{i_2} is sufficient. In addition, if any sk_i for some $i \notin \{i_1, i_2\}$ (obtained after the last rewinding for $\mathcal{B}$) is not valid, both $\mathcal{A}$ and $\mathcal{B}$ abort. Therefore, $\mathcal{A}$ and $\mathcal{B}$ behave identically, except if all secret keys sk_i (for $i \notin \{i_1, i_2\}$) are valid (after the last rewinding for $\mathcal{B}$) but $\mathcal{B}$ cannot find a valid key sk_{i_1} or a valid key sk_{i_2} among all keys it got from all the rewindings. We call 'bad case' the case where the previous bad event happens, and let ε_{bad} the probability of this event.

The advantage of $\mathcal{A}$ (to break the aggregator obliviousness) is exactly $Adv_{\mathcal{A}} = 2\varepsilon_{\mathcal{A}} - 1$, since when $\mathcal{A}$ plays against the real challenger for the aggregator obliviousness, $\mathcal{A}$ never aborts, and the bit b'' computed by $\mathcal{A}$ is always equal to b, the bit chosen in the game, hence $b' = b$ with probability $\varepsilon_{\mathcal{A}}$. In the bad case, the $\mathcal{B}$ outputs $b' = b$ with probability $1/2$ instead of $\varepsilon_{\mathcal{A}}$ for $\mathcal{A}$, while outside the bad case, $\mathcal{B}$ and $\mathcal{A}$ output $b' = b$ with the same probability. Therefore, when playing with $\mathcal{B}$, the reduction solves the original hard problem with probability at least:

$$\varepsilon_R - \varepsilon_{\text{bad}} \cdot |\varepsilon_{\mathcal{A}} - 1/2| = \varepsilon_R - \varepsilon_{\text{bad}} \cdot Adv_{\mathcal{A}}/2$$

which in term of advantage (if the reduction solves a decisional problem) is

$$2\left(\varepsilon_R - \varepsilon_{\text{bad}}\left|\varepsilon_{\mathcal{A}} - \tfrac{1}{2}\right|\right) - 1 = (2\varepsilon_R - 1) - \varepsilon_{\text{bad}}|2\varepsilon_{\mathcal{A}} - 1| = Adv_R - \varepsilon_{\text{bad}} \cdot Adv_{\mathcal{A}} \ .$$

This has to be negligible, otherwise the hard problem would not be hard. Therefore, ε_R or Adv_R cannot be larger than $\varepsilon_{\text{bad}} \cdot Adv_{\mathcal{A}}/2$ or $\varepsilon_{\text{bad}} \cdot Adv_{\mathcal{A}}$ (minus some negligible factor in the security parameter). We just need to prove that $\varepsilon_{\text{bad}} \geq 2/(n(n-1))$, to show our impossibility result.

For that purpose, let $E_{i_3,4}$ denote the event that, when the secret keys sk_i for $i \in \{1, \ldots, n\} \setminus \{i_3, i_4\}$ are corrupted, the secret keys given by the reduction are all valid. We clearly have:

$$\varepsilon_{\text{bad}} \leq \Pr\left[E_{i_1,i_2} \wedge \left(\bigwedge_{\{i_3,i_4\} \neq \{i_1,i_2\}} \neg E_{i_3,i_4}\right)\right] ,$$

since if E_{i_3,i_4} is true for some $\{i_3, i_4\} \neq \{i_1, i_2\}$ ($i_3 \neq i_4$), then $i_1 \notin \{i_3, i_4\}$ or $i_2 \notin \{i_3, i_4\}$, and we get a valid secret key sk_{i_1} or sk_{i_2}. If we fix everything except the choice of i_1 and i_2, each event E_{i_3,i_4} is either satisfied (it has probability 1) or not satisfied (it has probability 0). If two distinct event E_{i_5,i_6} and $E_{i'_5,i'_6}$ are satisfied, the event $\bigwedge_{\{i_3,i_4\} \neq \{i_1,i_2\}} \neg E_{i_3,i_4}$ never happens and $\varepsilon_{\text{bad}} = 0$. If no event is satisfied, the event E_{i_1,i_2} never happens and $\varepsilon_{\text{bad}} = 0$. Finally, if only one event E_{i_5,i_6} is satisfied, $E_{i_1,i_2} \wedge (\bigwedge_{\{i_3,i_4\} \neq \{i_1,i_2\}} \neg E_{i_3,i_4})$ happens only when $\{i_1, i_2\} = \{i_5, i_6\}$, which happens with probability $n(n-1)/2$. Therefore, $\varepsilon_{\text{bad}} \leq n(n-1)/2$.