On the Privacy of Concealed Data Aggregation[⋆]

Aldar C-F. Chan[1⋆⋆], Claude Castelluccia[2]

[1] School of Computing, National University of Singapore
aldar@comp.nus.edu.sg
[2] INRIA Rhône-Alpes, France
Claude.Castelluccia@inria.fr

Abstract. A formal treatment to the privacy of concealed data aggregation (CDA) is given. While there exist a handful of constructions, rigorous security models and analyses for CDA are still lacking. Standard security notions for public key encryption schemes, including semantic security and indistinguishability against chosen ciphertext attacks, are refined to cover the multi-sender nature and aggregation functionality of CDA in the security model. A generic CDA construction based on public key homomorphic encryption is given, along with a proof of its security in the proposed model. The security of two existing schemes is also analyzed in the proposed model.

1 Introduction

Concealed data aggregation (CDA) in which multiple source nodes send encrypted data to a sink along a concast tree with ciphertext aggregation performed en route is an active research problem, particularly in sensor networks [1, 3, 10, 26]. Privacy and message authentication are the two main security goals of CDA. This work focuses on the security model for privacy of CDA.

The privacy goal is two-fold. First, the privacy of the data has to be guaranteed end-to-end, that is, only the sink could learn about the final aggregation result and only a negligible amount of information about the final aggregate should be leaked out to any eavesdropper or node along the path. Each node should only have knowledge about its data, but no information about the data of other nodes. Second, to reduce communication overhead, the data from different source nodes have to be efficiently combined by intermediate nodes (i.e. aggregation) along the path. Nevertheless, these intermediate nodes should not learn any information about the final aggregate in an ideal scheme. It appears that these two goals are in conflict. As a result, deliberate study on the security definitions and rigorous analyses on CDA schemes are necessary. While there

⋆ The work described in this paper is based on results of IST FP6 STREP UbiSec&Sens. UbiSec&Sens receives research funding from the European Commission's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

⋆⋆ Part of this work was done while Aldar C-F. Chan was at INRIA.

are a handful of CDA constructions [1, 3, 10, 26] achieving various levels of privacy-efficiency tradeoff, a rigorous treatment to the security definitions, notions and analyses of CDA is still lacking. This work aims to fill the gap.

While there has been a solid foundation in cryptography for both private-key [23, 17, 16] and public-key [13, 20, 5, 12] encryption, a refinement to the standard security models is needed to cover the salient features in the CDA scenario: First, a CDA scheme can be based on private key or public key cryptography. That is, the encryption function of a CDA scheme could be public or private. Second, CDA is a many-to-one (multi-sender single-receiver) cryptosystem while cryptosystems in the literature are either one-to-one [16, 13] or one-to-many [24, 8]. Third, CDA includes the aggregation functionality on encrypted data whose adversary model needs a new definition. In this paper, we extend the standard security notions of semantic security and indistinguishability against chosen-ciphertext attacks to the CDA setting and analyze existing schemes [3, 26].

1.1 Related Work

Westhoff et. al gave the first CDA construction in [26, 10] based on the Domingo-Ferrer private key homomorphic encryption [6] and coined the term CDA. The scheme allows additive aggregation. Castelluccia et. al [3] constructed a stream cipher like CDA scheme for additive aggregation. In [1], Westhoff et. al. gave a private aggregation scheme for comparing encrypted data; however, the security of the proposed scheme is not reasonably high. It is fair to say that, despite the existence of these CDA constructions, a rigorous security model and analysis for CDA are still missing in the literature.

1.2 Our Contributions

The main contribution of this paper is the formalization of CDA. We extend the standard security notions of encryption schemes to cover the CDA scenario. Our security model covers both private-key and public-key based CDA constructions and takes into account the possibility of insider attacks due to compromised source nodes, as compared to [26, 10] which do not explicitly consider the threat of compromised nodes. It also includes the case in which the global randomness for encryption is prescribed beforehand or chosen by the sink and broadcast to the source nodes [3].

We also give a generic CDA construction based on any public key homomorphic encryption scheme. Provided that the underlying homomorphic encryption scheme is semantically secure, the CDA construction achieves semantic security against any coalition with up to $n-1$ compromised nodes where n is the total number of nodes in the system.[3]

Based on the CDA security model proposed in this paper, we analyze two existing schemes, namely, WGA [26] and CMT [3]. We show that WGA is only secure when there is no compromised node. Whereas, if the underlying pseudorandom function family (used for key generation) is (computationally) indistinguishable from a truly random

[3] In a general scenario, not all of the n nodes need to report in a given slot; only a subset of the n nodes contribute to the final aggregate. Without loss of generality, we assume all the n nodes contribute in the aggregation in the following discussion.

function, CMT can be proven to be semantically secure even when there are $n-1$ compromised nodes. For the pseudorandom function assumption to be held, it appears that a larger modulus size is needed as compared to that used in the original scheme. As an alternative, a hash variant of CMT which does not require a revision on the modulus size is given. Security preserves in the hashed variant if, given a uniformly distributed input, the hash function output follows a uniform distribution.

The rest of the paper is organized as follows. We give a brief introduction to the notations used in this paper in the next section. The definition of CDA and related security notions are presented in Sections 3 and 4 respectively. In Section 5, a generic CDA construction is given. The security of two existing schemes is analyzed in Section 6. We conclude the paper in Section 7.

2 Notations

We follow the notations for algorithms and probabilistic experiments that originate in [14]. A detailed exposition can be found there. We denote by $z \leftarrow A(x, y, \ldots)$ the experiment of running probabilistic algorithm A on inputs $x, y \ldots$, generating output z. We denote by $\{A(x, y, \ldots)\}$ the probability distribution induced by the output of A. The notations $x \leftarrow \mathcal{D}$ and $x \in_R \mathcal{D}$ are equivalent and mean randomly picking a sample x from the probability distribution $\mathcal{D}$; if no probability function is specified for $\mathcal{D}$, we assume x is uniformly picked from the sample space. We denote by $\mathbb{N}$ the set of non-negative integers. As usual, PPT denote probabilistic polynomial time. An empty set is always denoted by ϕ.

3 Definitions

A typical CDA scheme includes a sink R and a set U of n source nodes (which are usually sensor nodes) where $U = \{s_i : 1 \leq i \leq n\}$. Denote the set of source identities by ID; in the simplest case, $ID = [1, n]$. In the following discussion, $hdr \subseteq ID$ is a header indicating the source nodes contributing to an encrypted aggregate. Given a security parameter λ, a CDA scheme consists of the following polynomial time algorithms.

Key Generation (KG). Let $\mathsf{KG}(1^\lambda, n) \rightarrow (dk, ek_1, ek_2, \ldots, ek_n)$ be a probabilistic algorithm. Then, ek_i (with $1 \leq i \leq n$) is the encryption key assigned to source node s_i and dk is the corresponding decryption key given to the sink R.

Encryption (E). $\mathsf{E}_{ek_i}(m_i) \rightarrow (hdr_i, c_i)$ is a probabilistic encryption algorithm taking a plaintext m_i and an encryption key ek_i as input to generate a ciphertext c_i and a header $hdr_i \subset ID$. Here hdr_i indicates the identity of the source node performing the encryption; if the identity is i, then $hdr_i = \{i\}$.
We sometimes denote the encryption function by $\mathsf{E}_{ek_i}(m_i; r)$ to explicitly show by a string r the random coins used in the encryption process.

Decryption (D). Given an encrypted aggregate c and its header $hdr \subseteq ID$ (which indicates the source nodes included in the aggregation), $\mathsf{D}_{dk}(hdr, c) \rightarrow m/\perp$ is a deterministic algorithm which takes the decryption key dk, hdr and c as inputs and returns the plaintext aggregate m or possibly $\perp$ if c is an invalid ciphertext.

Aggregation (Agg). With a specified aggregation function f, the aggregation algorithm $\mathsf{Agg}_f(hdr_i, hdr_j, c_i, c_j) \rightarrow (hdr_l, c_l)$ aggregates two encrypted aggregates c_i and c_j with headers hdr_i and hdr_j respectively (where $hdr_i \cap hdr_j = \phi$) to create a combined aggregate c_l and a new header $hdr_l = hdr_i \cup hdr_j$. Suppose c_i and c_j are the ciphertexts for plaintext aggregates m_i and m_j respectively. The output c_l is the ciphertext for the aggregate $f(m_i, m_j)$, namely, $\mathsf{D}_{dk}(hdr_l, c_l) \rightarrow f(m_i, m_j)$. *Note that the aggregation algorithm does not need the decryption key dk or any of the encryption keys ek_i as input; it is a public algorithm.*

Depending on constructions, the aggregation function f could be any associative function, for instance, f could be the sum, multiplicative product, max, etc.. Leveraging on the associativity property, we abuse the notation in this paper: we denote the composition of multiple copies of f simply by $f(m_1, m_2, \ldots, m_i)$ irrespective of the order of aggregation and call it the f-aggregate on $m_1, m_2, \ldots, m_i$; to be precise, it should be written as $f(f(f(m_1, m_2), \ldots), m_i)$ with a certain aggregation order.

It is intentional to include the description of the header hdr in the above definition so as to make the CDA security model as general as possible (to cover schemes requiring headers in their operations). Nonetheless, generating headers or including headers as input to algorithms should not be treated as a requirement in the actual construction or implementation of CDA algorithms. For constructions which do not need headers (such as the generic construction given in Section 5), all hdr's can simply be treated as the empty set ϕ in the security model and the discussions in this paper still apply.

Typical CDA Operation. The operation of CDA runs as follows. In the initialization stage, the sink R runs KG to generate a set of encryption keys $\{ek_i : 1 \leq i \leq n\}$ and the corresponding decryption key dk and distributes each one of the encryption keys to the corresponding source, say, ek_i to s_i. Depending on constructions, the encryption keys ek_i could be private or public, but the decryption key dk has to be private in all cases.

At a certain instant, the sink selects a subset $S \subseteq U$ of the n sources to report their data. Each $s_i \in S$ uses its encryption key ek_i to encrypt its data represented by the plaintext m_i, giving a ciphertext c_i. We do not pose restrictions on whether global or local random coins should be used for encryption. If each source generates its random coins individually, the random coins are said to be local; if the random coins are chosen by the sink and broadcast to all source nodes, they are global. Global random coins are usually public. When global random coins are used, we do not pose restriction on the reuse of randomness despite that, in practice, each global random coin is treated as nonce, that is, used once only. The generic construction given in Section 5 uses local random coins whereas the CMT scheme [3] uses a global nonce.

Usually, the source nodes form a concast tree over which the encrypted data are sent. In order to save communication cost, aggregation is done en route to the sink whenever possible. When a node s_i in the tree receives x ciphertexts, say $(hrd_{i_1}, c_{i_1}), \ldots, (hdr_{i_x}, c_{i_x})$, from its children nodes[4] (with identities $i_1, \ldots, i_x \in S$), it aggregates these ciphertexts along with its own ciphertext (hdr_i, c_i) by running Agg_f successively. The concast tree structure ensures that any pair of these headers have an empty intersection. Suppose $c_{i_1}, \ldots, c_{i_x}$ are the ciphertexts for the plaintext aggregates $m_{i_1}, \ldots, m_{i_x}$. The resulting ciphertext is: (hdr_l, c_l) where $hdr_l = hdr_{i_1} \cup \ldots \cup hdr_{i_x} \cup hdr_i$ and c_l is the encryption of the aggregate $f(m_{i_1}, \ldots, m_{i_x}, m_i)$.

Eventually, a number of encrypted aggregates will arrive at the sink which combines them through running Agg_f to obtain a single encrypted aggregate c_{sink} and then applies the decryption algorithm to c_{sink} to get back the plaintext aggregate $f(\ldots, m_i, \ldots)$ with $s_i \in S$. We require the CDA be *correct* in the sense that when the encryption and decryption are performed with matched keys and correct headers and all the aggregations are run properly, the decryption should give back an f-aggregate of all the data applied to the encryption.

4 Security Notions

Two types of oracle queries (adversary interaction with the system) are allowed in the security model, namely, the encryption oracle $\mathcal{O}_E$ and the decryption oracle $\mathcal{O}_D$. Their details are as follows:

Encryption Oracle $\mathcal{O}_E(i, m)$. For fixed encryption and decryption keys, on input an encryption query $\langle i, m \rangle$, the encryption oracle retrieves s_i's encryption key ek_i and runs the encryption algorithm on m and replies with the ciphertext $\mathsf{E}_{ek_i}(m; r)$ and its header hdr. In case global random coins are used, the random coins r are part of the query input to $\mathcal{O}_E$.

Decryption Oracle $\mathcal{O}_D(hdr, c)$. For fixed encryption and decryption keys, on input a decryption query $\langle hdr, c \rangle$ (where $hdr \subseteq ID$), the decryption oracle retrieves the decryption key dk and runs the decryption algorithm D and sends the result $\mathsf{D}_{dk}(hdr, c)$ as the reply.

The encryption oracle is needed in the security model since the encryption algorithm in some CDA could use private keys, for examples [3, 26]. In case the encryption algorithm does not use any secret information, an adversary can freely generate the ciphertext on any message of his choice without relying on the encryption oracle.

4.1 Security against Chosen Ciphertext Attacks (CCA)

To define security (more precisely, indistinguishability) against adaptive chosen ciphertext attacks ($\mathsf{IND\text{-}CCA}2$), we use the following game played between a challenger and an adversary, assuming there is a set U of n source nodes. If no PPT adversary, even in collusion with at most t compromised node (with $t < n$), can win the game with non-negligible advantage (as defined below), we say the CDA scheme is t-secure.[5]

[4] It is possible that some of these ciphertexts are already the encryption of aggregated data rather than the encryption of a single plaintext.

[5] The adversary is allowed to freely choose parameters n and t.

Definition 1. *A CDA scheme is t-secure (indistinguishable) against adaptive chosen ciphertext attacks if the advantage of winning the following game is negligible in the security parameter λ for all PPT adversaries.*

Collusion Choice. The adversary chooses to corrupt t source nodes. Denote the set of these t corrupted nodes and the set of their identities by S' and I' respectively.

Setup. The challenger runs KG to generate a decryption key dk and n encryption keys $\{ek_i : 1 \leq i \leq n\}$, and gives the subset of t encryption keys $\{ek_j : s_j \in S'\}$ to the adversary but keeps the decryption key dk and the other $n - t$ encryption keys $\{ek_j : s_j \in U \backslash S'\}$.

Query 1. The adversary can issue to the challenger two types of queries:[6]

– Encryption Query $\langle i_j, m_j \rangle$. The challenger responds with $\mathsf{E}_{e_{i_j}}(m_j)$.

– Decryption Query $\langle hdr_j, c_j \rangle$. The challenger responds with $\mathsf{D}_{dk}(hdr_j, c_j)$.

Challenge. Once the adversary decides that the first query phase is over, it selects a subset S of d source nodes (whose identities are in the set I) such that $|S \backslash S'| > 0$, and outputs two different sets of plaintexts $M_0 = \{m_{0k} : k \in I\}$ and $M_1 = \{m_{1k} : k \in I\}$ to be challenged. The only constraint is that the two resulting plaintext aggregates x_0 and x_1 are not equal where $x_0 = f(\ldots, m_{0k}, \ldots)$ and $x_1 = f(\ldots, m_{1k}, \ldots)$.

The challenger flips a coin $b \in \{0, 1\}$ to select between x_0 and x_1. The challenger then encrypts[7] each $m_{bk} \in M_b$ with ek_k and aggregates the resulting ciphertexts in the set $\{\mathsf{E}_{ek_k}(m_{bk}) : k \in I\}$ to form the ciphertext C of the aggregate, that is, $\mathsf{D}_{dk}(I, C) = x_b$, and gives (I, C) as a challenge to the adversary.

Query 2. The adversary is allowed to make more queries (both encryption and decryption) as previously done in Query 1 phase but no decryption query can be made on the challenged ciphertext C. Nevertheless, the adversary can still make a decryption query on a header corresponding to the set S except that the ciphertext has to be chosen different from the challenged ciphertext C.

Guess. Finally, the adversary outputs a guess $b' \in \{0, 1\}$ for b.

Result. The adversary wins the game if $b' = b$. The advantage of the adversary is defined as: $Adv_{\mathcal{A}} = \left| Pr[b' = b] - \frac{1}{2} \right|$.

Note that in CDA what the adversary is interested in is the information about the final aggregate. Consequently, in the above game, the adversary is asked to distinguish between the ciphertexts of two *different* aggregates x_0 and x_1 as the challenge, rather than to distinguish two different sets of plaintexts M_0 and M_1. By picking elements for M_0 and M_1, the adversary is essentially free to choose x_0 and x_1. Allowing the adversary to choose the two sets M_0, M_1 is to give him more flexibility in launching attacks. When an adversary cannot distinguish between the ciphertexts of two different aggregates (of his choice) with probability of success non-negligibly greater than $1/2$, this means, in essence, he can learn no information about an aggregate from its ciphertext.

[6] In case global random coins are used, the adversary is allowed to choose and submit his choices of random coins for both encryption and decryption queries. Depending on whether the encryption keys are kept secret, the encryption queries may or may not be needed.

[7] In case global random coins are used for encryption, the challenger chooses and passes them to the adversary. If a *nonce* is used, the global random coins should be chosen different from those used in the Query 1 phase and no query on them should be allowed in the Query 2 phase.

4.2 Semantic Security

Semantic security, which is equivalent to indistinguishability against chosen plaintext attacks (IND-CPA), is defined by the same game as in the definition of security against chosen ciphertext attacks in Section 4.1 except that no query to the decryption oracle $\mathcal{O}_D$ is allowed. Similar to the definition in Section 4.1, a CDA scheme is said to be t-secure when it can still achieve semantic security against a PPT adversary corrupting at most t compromised nodes.

For a CDA scheme to be useful, it should at least achieve semantic security. In the notion of semantic security, the main resource for an adversary is the encryption oracle $\mathcal{O}_E$. In some schemes like [26, 3], the adversary may not know the encryption keys, meaning he might not have access to the encryption oracle in the real environment. Nevertheless, in sensor networks, he is able to obtain the encryption of any plaintext of his choice by manipulating the sensing environment and recording the sensed value using his own sensors. Hence, chosen plaintext attacks are still a real threat to CDA.

4.3 One-wayness

One-wayness is the weakest possible security notion for encryption. A CDA scheme is t-secure in one-wayness if no PPT attacker, corrupting at most t nodes, should be able, with non-negligible probability of success, to recover the plaintext aggregate matching a given ciphertext. To define one-wayness more formally, we can use the same game in Section 4.1 except that no query is allowed and the adversary can make no choice in the challenge phase but is given a ciphertext of a certain aggregate x (encrypted using at least one encryption key not held by the adversary) and asked to recover x.

5 A Generic CDA Construction

In this section, a generic construction of semantically secure CDA (using local random coins) is given based on any semantically secure public-key homomorphic encryption. The result is not surprising but could be useful. Note that an asymmetric key homomorphic encryption is used in this construction, compared to the symmetric key encryption used in the WGA construction [26]. An asymmetric key encryption is necessary in order to guard against possible insider attacks from compromised nodes.

5.1 Public Key Homomorphic Encryption

A public key homomorphic encryption scheme is a 4-tuple (KG, E, D, A). The key generation algorithm KG receives the security parameter 1^λ as input and outputs a pair of public and private keys (pk, sk). E and D are the encryption and decryption algorithms. Given a plaintext x and random coins r, the ciphertext is $E_{pk}(x; r)$ and $D_{sk}(E_{pk}(x; r)) = x$. The homomorphic property allows one to operate on the ciphertexts using the poly-time algorithm A without first decrypting them; more specifically, for any x, y, r_x, r_y, A can generate from $E_{pk}(x; r_x)$ and $E_{pk}(y; r_y)$ a new ciphertext of the form $E_{pk}(x \otimes y; s)$ for some s. The operator $\otimes$ could be addition, multiplication or others depending on specific schemes; for instance, it is multiplication for RSA [22] or ElGamal [7] and addition for Paillier [21].

As observed in previous work in the literature, due to the homomorphic property, achieving IND-CCA2 security could be impossible for homomorphic encryption. The notion of security against CCA1 attacks is not often considered in practical constructions. Hence, semantic security or the equivalent notion of IND-CPA security appears to be the de facto security notion for homomorphic encryption schemes. In brief, the IND-CPA notion can be described by the following game: in the Setup phase, the challenger runs $KG(1^\lambda)$ to generate a pair of public and private keys, gives the public key to the adversary but keeps the private key. The adversary can freely encrypt any message of his choice using the public key. The adversary chooses two different messages m_0, m_1 and gives them to the challenger which flips a coin $b \in \{0, 1\}$ and gives $E_{pk}(m_b; r)$ to the adversary. The adversary has to output a guess b' for b and his advantage of winning the game is defined as $\left|Pr[b' = b] - \frac{1}{2}\right|$. If the advantage of winning the above game is negligible in the security parameter λ for all PPT adversaries, then the scheme is IND-CPA secure.

5.2 Concealed Data Aggregation from Public Key Homomorphic Encryption

Assume there are n source nodes in total. Suppose there exists a semantically secure public-key homomorphic encryption scheme $(KG^{HE}, E^{HE}, D^{HE}, A^{HE})$ with homomorphism on operator $\otimes$. We can construct a semantically secure CDA scheme, tolerating up to $n - 1$ compromised nodes, with aggregation function of the form: $f(m_i, m_j) = m_i \otimes m_j$. The construction is as follows: (The headers are included in the following description for completeness; they are not needed in the construction. In fact, all these hdr_i's are the empty set ϕ.)

Key Generation (KG). Run $KG^{HE}(1^\lambda)$ to generate (pk, sk). Set the CDA decryption key $dk = sk$ and each one of the CDA encryption keys to be pk, that is, $ek_i = pk, \forall i \in [1, n]$.

Encryption (E). Given a plaintext data m_i, toss the random coins r_i needed for E^{HE} and output $c_i = E^{HE}_{pk}(m_i; r_i)$. Set the header $hdr_i = \phi$. Output (hdr_i, c_i).

Decryption (D). Given an encrypted aggregate c and its header hdr, run D^{HE} using the private key sk to decrypt c and output $x = D^{HE}_{sk}(c)$ as the plaintext aggregate.

Aggregation (Agg). Given two CDA ciphertexts (hdr_i, c_i) and (hdr_j, c_j), the aggregation can be done using the homomorphic property of the encryption scheme. Generate $c_l = A^{HE}(c_i, c_j)$ and $hdr_l = hdr_i \cup hdr_j$. Output (hdr_l, c_l).

Correctness. Without loss of generality, we consider the case with only two plaintext messages m_i and m_j and ignore the header part as it is always equal to ϕ. The corresponding ciphertexts for m_i and m_j are $c_i = E^{HE}_{pk}(m_i; r_i)$ and $c_j = E^{HE}_{pk}(m_j; r_j)$ for some random coins r_i, r_j. If the aggregation is done using Agg as described above, the aggregation result c_l should be equal to $E^{HE}_{pk}(m_i \otimes m_j; s)$ for some s. In essence, this value is $E^{HE}_{pk}(f(m_i, m_j), s)$. With the correctness property of the homomorphic encryption scheme, $D^{HE}_{sk}(c_l)$ should give back $m_i \otimes m_j$ which is the aggregate $f(m_i, m_j)$.

The security of the CDA construction is best described by the following theorem.

Theorem 1. *For a total of n source nodes, the above CDA construction is semantically secure against any collusion of at most $n-1$ compromised nodes, assuming that the underlying homomorphic encryption scheme is semantically secure. The advantage for any PPT adversary in breaking the semantic security of the CDA construction is bounded above by the advantage achievable (by all PPT adversaries) in breaking the semantic security of the underlying homomorphic encryption scheme.*

Proof. It is trivial that security against $n-1$ compromised nodes implies security against $t < n-1$ compromised nodes, and the advantages are related by a constant factor with respect to λ. Hence, we consider the case with $n-1$ compromised nodes.

We prove by contradiction. Assume the underlying homomorphic encryption is semantically secure, that is, all PPT algorithms have negligible advantage to break the semantic security of the scheme. Suppose there exists a PPT adversary $\mathcal{A}$ which, in coalition with $n-1$ nodes, can break the semantic security property of the CDA construction with non-negligible advantage. We show how to use $\mathcal{A}$ to construct another algorithm $\mathcal{A}'$ to break the semantic security of the homomorphic encryption as follows:

Algorithm $\mathcal{A}'$

Setup. Receive the public key pk from the challenger and pass it to the n source nodes. Allow the adversary $\mathcal{A}$ to choose any $n-1$ nodes to corrupt.

Query. Since no private key is needed for encryption, no $\mathcal{O}_E$ query is necessary.

Challenge. In the challenge phase, receive from $\mathcal{A}$ two sets of plaintext messages $M_0 = \{m_{01}, m_{02}, \ldots, m_{0n}\}$ and $M_1 = \{m_{11}, m_{12}, \ldots, m_{1n}\}$. Since $\mathcal{A}$ has corrupted $n-1$ nodes, $|M_0|$ and $|M_1|$ have to be equal to n. Compute $x_0 = f(m_{01}, m_{02}, \ldots, m_{0n})$ and $x_1 = f(m_{11}, m_{12}, \ldots, m_{1n})$ and output x_0, x_1 to the challenger for a challenged ciphertext c. (Note that the constraint posed on the challenge in Definition 1 in Section 4.1 assures that $x_0 \neq x_1$.)

Guess. Let the challenged ciphertext $c = E^{HE}_{pk}(x_b; r)$ for some unknown random coins r where $b \in \{0,1\}$ is unknown. Pass c as the challenge for $\mathcal{A}$. When $\mathcal{A}$ outputs b', output b' as a guess for b to the challenger.

In the above simulation, the challenge c is generated by first aggregating the plaintext and then encrypting the plaintext aggregate with some random coins r. In a real attack, each $m_{bi} \in M_b$ is encrypted with some random coins r_i and the resulting ciphertexts are then aggregated to generate c, which in essence is the ciphertext for the plaintext aggregate encrypted with some random coins s whose relationship with r_i's is unknown. If these r_i's are independently picked at random, then the resulting randomness s would have the same distribution as a randomly picked r. Hence, the distributions of the challenge c generated by the two processes are indistinguishable. In other words, the view of the adversary $\mathcal{A}$ in the above simulation is essentially the same as that in a real attack.

Let $Adv^{\textsf{CDA-IND-CPA}}_{\mathcal{A}}(\lambda)$ be the advantage of the adversary $\mathcal{A}$ in breaking the semantic security of the CDA construction. The advantage $Adv^{\textsf{HE-IND-CPA}}_{\mathcal{A}'}(\lambda)$ of $\mathcal{A}'$ in breaking the semantic security of the underlying homomorphic encryption is then:

$$Adv^{\textsf{HE-IND-CPA}}_{\mathcal{A}'}(\lambda) = Adv^{\textsf{CDA-IND-CPA}}_{\mathcal{A}}(\lambda).$$

If $Adv_{\mathcal{A}}^{\textsf{CDA-IND-CPA}}(\lambda)$ is non-negligible, so is $Adv_{\mathcal{A}'}^{\textsf{HE-IND-CPA}}(\lambda)$ (a contradiction). $\square$

6 Security Analysis of Existing Schemes

In this section, we analyze two practical schemes in the literature in the proposed security model, and propose modifications to one of them in Section 6.3.

6.1 WGA [26]

WGA uses Domingo-Ferrer's symmetric-key homomorphic encryption as a building block. Each source node uses the same encryption key ek and the sink's decryption key $dk = ek$. When there is no compromised node, if the underlying symmetric-key homomorphic encryption is semantically secure, then WGA achieves semantic security. The analysis is straightforward. Suppose there is an adversary $\mathcal{A}$ which can break the semantic security of WGA. It is trivial that $\mathcal{A}$ can be used as a subroutine of another algorithm $\mathcal{A}'$ to break the semantic security of the underlying encryption. Besides, any encryption oracle query from $\mathcal{A}$ can be answered easily by $\mathcal{A}'$ using the query result from the challenger of the underlying encryption scheme; in other words, the view to $\mathcal{A}$ in this simulation is indistinguishable from that in the real attack.

However, as few as one node is compromised, the adversary knows the decryption key and can gain the knowledge of all future aggregates by just passive eavesdropping, that is, not even one-wayness can be achieved if there exists compromised nodes.

6.2 CMT [3]

CMT can be considered as a practical modification of the Vernam cipher or one-time pad [25] to allow plaintext addition to be done in the ciphertext domain. Basically, there are two modifications. First, the exclusive-OR operation is replaced by an addition operation. By choosing a proper modulus, multiplicative aggregation is also possible in CMT.[8] Second, instead of uniformly picking a key at random from the key space, the key is generated by a certain deterministic algorithm (with an unknown seed) such as a pseudorandom function [11]. As a result, the information-theoretic security (which requires the key be at least as long as the plaintext) in the Vernam cipher is replaced with a security guarantee in the computational-complexity theoretic setting in CMT.

The operation of the CMT scheme is as follows: (The description could be slightly different from the original scheme [3] as the procedures to generate the encryption keys from a pseudorandom function are filled in.) Let p be a large enough integer used as the modulus. Assume the key length is λ bits. Then p could be 2^λ. Besides, global random coins are used in CMT, that is, the sink chooses and broadcasts a public nonce to all nodes.

In the following description, let $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a pseudorandom function family where $F_\lambda = \{f_s : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a collection of functions indexed by a key $s \in \{0,1\}^\lambda$. For details on pseudorandom functions, [11] has a comprehensive description. Loosely speaking, given a function f_s from a pseudorandom function

[8] CMT can achieve either additive or multiplicative aggregation but not both at the same time.

ensemble with unknown key s, any PPT distinguishing procedure allowed to get the values of $f_s(\cdot)$ at (polynomially many) arguments of its choice should not be able to tell (with non-negligible advantage in λ) whether the answer of a new query (with the argument not queried before) is supplied by f_s or randomly picked from $\{0,1\}^\lambda$.

Key Generation (KG). Randomly pick $K \in \{0,1\}^\lambda$ and set it as the decryption key dk. For each $i \in [1,n]$, $ek_i = f_K(i)$ is the encryption key for source node s_i with identity i.

Encryption (E). Given an encryption key ek_i, a plaintext data m_i and a broadcast nonce r from the sink, output $c_i = (m_i + f_{ek_i}(r)) \bmod p$. Set the header $hdr_i = \{i\}$. Output (hdr_i, c_i). *Note: each r has to be used once only.*

Decryption (D). Given the ciphertext (hdr, c) of an aggregate and a nonce r used in the encryption, generate $ek_i = f_K(i), \forall i \in hdr$. Output the plaintext aggregate $x = (c - \sum_{i \in hdr} f_{ek_i}(r)) \bmod p$.

Aggregation (Agg). Given two CDA ciphertexts (hdr_i, c_i) and (hdr_j, c_j), compute $c_l = (c_i + c_j) \bmod p$ and $hdr_l = hdr_i \cup hdr_j$ and output (hdr_l, c_l).

How good the CMT scheme achieves IND-CPA security relies on how good the underlying key generation function is as a pseudorandom function. As a consequence, the required modulus size is determined mainly by the parameters of the conjectured pseudorandom function family used, rather than the size of the largest plaintext aggregate. There are various constructions of pseudorandom functions [18, 19, 15, 2], each of which is based on a different computational assumption and requires different computational resources; it is therefore difficult to evaluate the efficiency of the CMT scheme without seeing the actual implementation. The security of the CMT can be summarized by the following theorem.

Theorem 2. *The CMT scheme is semantically secure against any collusion with at most $n-1$ compromised nodes, assuming $F_\lambda = \{f_s : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a pseudorandom function.*

Proof. Without loss of generality, we prove the security of a modified version of CMT in which each encryption key is uniformly picked from $\{0,1\}^\lambda$, compared with keys generated by a pseudorandom function in the actual CMT scheme. We then provide a justification why the inference applies to the actual CMT implementation.

Indistinguishability Property of a Pseudorandom Function. Assume f is taken from a pseudorandom function. Then for a fixed input argument x and and an unknown, randomly picked key K, the following two distributions are computationally indistinguishable provided that polynomially many (say q) evaluations of $f_K(\cdot)$ have been queried:

$$\{y = f_K(x) : y\}, \{y \leftarrow \{0,1\}^\lambda : y\}.$$

That is, the output $f_K(x)$ is computationally indistinguishable from a randomly picked number from $\{0,1\}^\lambda$ to any PPT distinguisher who has knowledge of the input argument x and a set of polynomially many 2-tuples $(x_i, f_K(x_i))$ where $x_i \neq x$. More formally, for any PPT distinguisher $\mathcal{D}$,

$$|Pr[y = f_K(x) : \mathcal{D}(x,y) = 1] - Pr[y \leftarrow \{0,1\}^\lambda : \mathcal{D}(x,y) = 1]| < \varepsilon(\lambda)$$

where $\varepsilon(\lambda)$ is a negligible function in λ.

Suppose there exists a PPT adversary D which can break the semantic security of CMT with non-negligible advantage Adv_D^{CMT}. We show in the following how D can be used to construct an algorithm D' which can distinguish the above distributions with non-negligible advantage. Assume the key K in question is unknown to D'.

Algorithm D'

Setup. Allow the adversary D to choose any $n-1$ sources to corrupt. Randomly pick $n-1$ encryption keys $ek_i \in_R \{0,1\}^\lambda$ and pass them to the adversary. Assume node n is uncorrupted. The encryption key for node n is taken to be K, the key of the pseudorandom function D' is being challenged with. That is, K is unknown to D'.

Query. Upon receiving an encryption query $\langle i_j, m_j\rangle$ with nonce r_j, return $c_j = (f_{ek_{i_j}}(r_j) + m_j) \bmod p$ if $i_j \neq n$. Otherwise, pass r_j to query the pseudorandom function to get back $f_K(r_j)$ and reply with $c_j = (f_K(r_j) + m_j) \bmod p$.

Challenge. In the challenge phase, receive from D two sets of plaintext messages $M_0 = \{m_{01}, m_{02}, \ldots, m_{0n}\}$ and $M_1 = \{m_{11}, m_{12}, \ldots, m_{1n}\}$.
Randomly pick a number w and output it to the pseudorandom function challenger to ask for a challenge. Note w is the nonce used for CDA encryption in the challenge for D. The pseudorandom function challenger flips a coin $b \in \{0,1\}$ and returns t_b, which is $f_K(w)$ when $b=0$ and randomly picked from $\{0,1\}^\lambda$ when $b=1$. These two cases corresponds to the two distributions discussed above.
Randomly flip a coin $d \in \{0,1\}$, and return the challenge ciphertext c_d to D where $c_d = \sum_{i=1}^n m_{di} + \sum_{i=1}^{n-1} f_{ek_i}(w) + t_b$.

Guess. D returns its guess b'. Return b'' which is 0 when $b' = d$ and 1 otherwise.

Obviously, if D is PPT, then D' is also PPT. Denoting the expression $\sum_{i=1}^n m_{di} + \sum_{i=1}^{n-1} f_{ek_i}(w)$ by X_d, the challenge passed to D can be expressed as $c_d = X_d + t_b$. When $b=0$, $t_b = f_K(w)$; when $b=1$, t_b is a randomly picked number from $\{0,1\}^\lambda$. In the following discussion, we denote the output of D on input c_d by $D(c_d)$. The probability of success for D' to distinguish between $f_K(w)$ and a random number is:

$$\begin{aligned}
Pr_{D'}^{PRF}[Success] &= Pr[b'' = b] \\
&= \tfrac{1}{2}\{Pr[b''=0|b=0] + Pr[b''=1|b=1]\} \\
&= \tfrac{1}{4}\{Pr[b''=0|b=0,d=0] + Pr[b''=0|b=0,d=1] \\
&\quad +Pr[b''=1|b=1,d=0] + Pr[b''=1|b=1,d=1]\} \\
&= \tfrac{1}{4}\{Pr[D(t_0+X_0)=0] + Pr[D(t_0+X_1)=1] \\
&\quad +Pr[D(t_1+X_0)=1] + Pr[D(t_1+X_1)=0]\} \\
&= \tfrac{1}{4}\{Pr[D(t_0+X_0)=0] + Pr[D(t_0+X_1)=1] \\
&\quad +1 + Pr[D(t_1+X_0)=1] - Pr[D(t_1+X_1)=1]\} \\
&= \tfrac{1}{4}\{2Pr_D^{CMT}[Success] + 1 \\
&\quad +(Pr[D(t_1+X_0)=1] - Pr[D(t_1+X_1)=1])\}.
\end{aligned}$$

Note that t_0+X_0 and t_0+X_1 are valid CMT ciphertexts for the two challenges plaintext sets M_0 and M_1 respectively. In the last step, we make use of the fact that the probability of success for D to break the semantic security of CMT is given by:

$$Pr_D^{CMT}[Success] = \frac{1}{2}Pr[D(t_0 + X_0) = 0] + \frac{1}{2}Pr[D(t_0 + X_1) = 1].$$

Rearranging terms, we have

$$\begin{array}{ll} 4Pr_{D'}^{PRF}[Success] & = 2Pr_D^{CMT}[Success] + 1 \\ +Pr[D(t_1 + X_1) = 1] - Pr[D(t_1 + X_0) = 1] & \\ 4(Pr_{D'}^{PRF}[Success] - \frac{1}{2}) & = 2(Pr_D^{CMT}[Success] - \frac{1}{2}). \\ +Pr[D(t_1 + X_1) = 1] - Pr[D(t_1 + X_0) = 1] & \end{array}$$

Taking absolute value on both sides and substitute $Adv_{D'}^{PRF} = |Pr_{D'}^{PRF}[Success] - \frac{1}{2}|$ and $Adv_D^{CMT} = |Pr_D^{CMT}[Success] - \frac{1}{2}|$, we have

$$2Adv_{D'}^{PRF} + \frac{1}{2}\left|Pr[D(t_1 + X_1) = 1] - Pr[D(t_1 + X_0) = 1]\right| \geq Adv_D^{CMT}.$$

Since t_1 is a randomly picked number, $\{t_1 + X_0\}$ and $\{t_1 + X_1\}$ are identically distributed. That is, for any PPT algorithm D, $Pr[D(t_1+X_0) = 1] = Pr[D(t_1+X_1) = 1]$. Hence,

$$2Adv_{D'}^{PRF}(\lambda) \geq Adv_D^{CMT}(\lambda).$$

Note also that:

$$\big|\, Pr[y = f_K(x) : D'(x, y) = 1] - Pr[y \leftarrow \{0, 1\}^\lambda : D'(x, y) = 1] \,\big| = 2Adv_{D'}^{PRF}(\lambda).^{9}$$

If Adv_D^{CMT} is non-negligible in λ, then so is $Adv_{D'}^{PRF}$. As a result, if D can break the semantic security of CMT with non-negligible advantage, D' could distinguish between the output of pseudorandom function f and a random number. Equivalently, $|Pr[y = f_K(x) : D'(x, y) = 1] - Pr[y \leftarrow \{0, 1\}^\lambda : D'(x, y) = 1]|$ is non-negligible (a contradiction to the indistinguishability property of a pseudorandom function).

The above security argument applies to the actual CMT implementation since the view of the adversary D in the above simulation is in essence the same as that in the actual CMT scheme. For each one of the $n-1$ corrupted node, the encryption key is $f_{K'}(i)$ ($1 \leq i \leq n - 1$) for some randomly picked master key K'. By the property of pseudorandom function, $f_{K'}(i)$ is indistinguishable from a randomly picked key (as used in the above simulation game) for all PPT distinguisher algorithms. For the uncorrupted node, its output for encryption is now $f_{f_{K'}(n)}(x)$ instead of $f_K(x)$ (with randomly picked K) as used in the above simulation game. It can be shown by a contrapositive argument that, for fixed n and given x, the two distributions are computationally indistinguishable, that is,

$$\{K' \leftarrow \{0, 1\}^\lambda : (x, f_{f_{K'}(n)}(x))\} \overset{c}{\equiv} \{K \leftarrow \{0, 1\}^\lambda : (x, f_K(x))\}.$$

[9] The derivation is as follows.
$$\begin{aligned} &\big|Pr[y = f_K(x) : D'(x, y) = 1] - Pr[y \leftarrow \{0, 1\}^\lambda : D'(x, y) = 1]\big| \\ &= \big|1 - Pr[y = f_K(x) : D'(x, y) = 0] - Pr[y \leftarrow \{0, 1\}^\lambda : D'(x, y) = 1]\big| \\ &= \big|1 - 2Pr_{D'}^{PRF}[Success]\big| \\ &= 2 \cdot \big|Pr_{D'}^{PRF}[Success] - \tfrac{1}{2}\big| \\ &= 2 \cdot Adv_{D'}^{PRF}(\lambda) \end{aligned}$$

The argument is as follows: Assume f is a pseudorandom function. That is, $A = \{K' \leftarrow \{0,1\}^\lambda : f_{K'}(n)\}$ is indistinguishable from $B = \{K \leftarrow \{0,1\}^\lambda : K\}$ for all PPT distinguishers. If there exists a PPT distinguisher D which can distinguish between $X = \{K' \leftarrow \{0,1\}^\lambda : (x, f_{f_{K'}(n)}(x))\}$ and $Y = \{K \leftarrow \{0,1\}^\lambda : (x, f_K(x))\}$, we can use D to distinguish between A and B. The idea is when we receive a challenge s which could be from A or B, we send x and $f_s(x)$ as a challenge for D. If s belongs to A, $(x, f_s(x))$ belongs to X, and if s belongs to B, $(x, f_s(x))$ belongs to Y. We could thus distinguish X from Y (a contradiction). □

6.3 A Hashed Variant of CMT

As discussed in the previous section, when pseudorandom functions are used to generate encryption keys for CMT, the modulus size has to be revised and the advantage of short ciphertext in CMT is lost. In order to maintain the same ciphertext size, the output of the pseudorandom function can be hashed down by some good hash function $h : \{0,1\}^\lambda \rightarrow \{0,1\}^l$ where λ is the security parameter for the pseudorandom function and l is the size of the maximum plaintext aggregate. Instead of using the output of the pseudorandom function directly for encryption, its hashed value is input to the encryption algorithm. For a given plaintext m_i, a nonce r and an encryption key e_i, the ciphertext of the hashed CMT is: $c_i = (m_i + h(f_{e_i}(r))) \bmod p'$ where $|p'| = l$. The decryption algorithm is modified accordingly to hash the output of the pseudorandom function and then subtract the hash values from the ciphertext.

Requirement on the Hash Function. In order to preserve semantic security for the hashed CMT scheme, the hash function $h : \{0,1\}^\lambda \rightarrow \{0,1\}^l$ needs to satisfy the following property: $\{t \leftarrow \{0,1\}^\lambda : h(t)\}$ has a uniform distribution over $\{0,1\}^l$.

We can actually view h as a length-compressing function which matches the output length of a pseudorandom function with the size of the modulus in use. While the idealized hash function in the random oracle model is sufficient to fulfill the above mentioned requirement, it is probably more than necessary.

Note that for an ideal pseudorandom function family, h might simply be implemented by truncating the pseudorandom function output to fit the modulus size. However, to take into account of the imperfectness of the conjectured pseudorandom function families used in practice, it could be preferable if the pseudorandom function output is divided into small segments which are then combined by taking exclusive OR. Of course, the output size of the pseudorandom function has to be a multiple of the modulus size to implement this approach.

Security of the Hashed CMT. Only a few modifications to the security proof in Section 6.2 are needed in order to prove the security of the hashed variant.

First, in the algorithm D', all cipertexts are now generated using the hashed values of the pseudorandom function outputs or replies from the challenger of D'. With such changes, we now denote the expression $\sum_{i=1}^{n} m_{di} + \sum_{i=1}^{n-1} h(f_{ek_i}(w))$ by X_d. Of course, the modulus size would be l instead of λ.

Second, the challenge passed to D would be: $c_d = X_d + h(t_b)$. Then the derivation for the advantage expressions is essentially the same as that for CMT.

Third, the security proof of CMT relies on the fact that $\{t_1 \leftarrow \{0,1\}^\lambda : t_1 + X_0\}$ and $\{t_1 \leftarrow \{0,1\}^\lambda : t_1 + X_1\}$ are identical distribution. On the contrary, to prove the security of hashed CMT, we need the following distributions to be identical:

$$\{t_1 \leftarrow \{0,1\}^\lambda : h(t_1) + X_0\}, \{t_1 \leftarrow \{0,1\}^\lambda : h(t_1) + X_1\}.$$

If h fulfills the requirement mentioned above, then $\{t_1 \leftarrow \{0,1\}^\lambda : h(t_1)\}$ is the uniform distribution over $\{0,1\}^l$. Consequently, the above two distributions are identical. This thus conclude the proof that hashed CMT is semantically secure.

The modification of the hash variant of CMT shares similarities with the hashed Diffie-Hellman scheme to get rid of the group encoding problem [4, 9] in the algebraic group used. While the hash function has to be modeled as a random oracle in order to prove the security of the hashed Diffie-Hellman scheme, the security proof of CMT applies to the hash variant of CMT without relying on the random oracle model. The main reason for the difference is: in the security proof for the hashed Diffie-Hellman scheme, the random oracle is used for answering queries to the decryption oracle, while in hashed CMT, no decryption oracle access is allowed in the security model as we only prove hashed CMT achieves semantic security.

7 Conclusions

In this paper, we give a rigorous treatment to the CDA problem. More specifically, we extend standard privacy notions to cover the CDA scenario which is a multiple-sender cryptosystem and supports aggregation. We also give a generic CDA construction based on any semantically secure public key encryption scheme and prove that it achieves semantic security. Besides, we analyze the security of two existing constructions, namely WGA and CMT, in the proposed model. We also propose a hashed variant of CMT to achieve security and efficiency simultaneously. As future work, we will study security model for aggregate authenticity; however, secure versions of the natural extension of MAC [2] (supporting message aggregation) may not exist. The reason is that if such a MAC scheme exists, it can be used to construct, from any semantically secure CDA, an IND-CCA2 secure CDA (which may not be achievable).

References

1. M. Acharya, J. Girao, and D. Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *the Proceedings of WiOpt 2005*, April 2005.
2. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology — CRYPTO 1996, Springer-Verlag LNCS vol. 1109*, pages 1–15.
3. C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *the Proceedings of MobiQuitous'05*, pages 1–9, July 2005.
4. B. Chevallier-Mames, P. Paillier, and D. Pointcheval. Encoding-free ElGamal encryption without random oracles. In *Public Key Cryptography (PKC 2006, Springer-Verlag LNCS vol. 3958*, pages 24–26, 2006.

5. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing,* 30(2):391–437, 2000.
6. J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *the Proceedings of 5th International Conference on Information Security (ISC'02) , Springer-Verlag LNCS vol. 2433,* pages 471–483, September 2002.
7. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-30(4):469–472, July 1985.
8. A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology — CRYPTO 1993, Springer-Verlag LNCS vol. 773,* pages 480–491, 1994.
9. R. Gennaro, H. Krawczyk, and T. Rabin. Secure hashed Diffie-Hellman over non-DDH groups. In *Advances in Cryptology — EUROCRYPT 2004, Springer-Verlag LNCS vol. 3027*, pages 361–381, 2004.
10. J. Girao, D. Westhoff, and M. Schneider. CDA: Concealed data aggregation in wireless sensor networks. In *the Proceedings of IEEE International Conference on Communication(ICC'05),* May 2005.
11. O. Goldreich. *Foundations of Cryptography: Part 1.* Cambridge University Press, 2001.
12. O. Goldreich. *Foundations of Cryptography: Part 2.* Cambridge University Press, 2004.
13. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences,* 28(2):270–299, 1984.
14. S. Goldwasser, S. Micali, and R. Rivest. A secure signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
15. T. Iwata and K. Kurosawa. OMAC: One-key CBC MAC. In *Fast Software Encryption (FSE 2003), Springer-Verlag LNCS vol. 2887,* pages 129–153, 2003.
16. J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology*, 19(1):67–95, 2006.
17. M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton, NJ, USA, 1996.
18. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *the Proceedings of IEEE Symposium on Foundations on Computer Science (FOCS'97)*, pages 458–467, 1997.
19. M. Naor, O. Reingold, and A. Rosen. Pseudorandom functions and factoring. *SIAM Journal on Computing,* 31(5):1383–1404, 2002.
20. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen-ciphertext attacks. In *ACM Symposium on Theory of Computing (STOC 1990)*, pages 427–437, 1990.
21. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT 1999, Springer-Verlag LNCS vol. 1592*, pages 223–238, 1999.
22. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of ACM*, 21(2):120–126, February 1978.
23. C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, 1949.
24. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, 2002.
25. G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of the American Institute of Electrical Engineers*, 45:105–115, 1926. See also US patent #1,310,719.
26. D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaption. *IEEE Transactions on Mobile Computing,* 5(10):1417–1431, 2006.