Journal of Ambient Intelligence and Humanized Computing (2020) 11:5103–5113
https://doi.org/10.1007/s12652-020-01814-5

ORIGINAL RESEARCH

Cheater-identifiable homomorphic secret sharing for outsourcing computations

Yan He[1,2,3] · Liang Feng Zhang[1]

Received: 7 September 2019 / Accepted: 18 February 2020 / Published online: 2 March 2020

Abstract
Homomorphic secret sharing (HSS) allows a dealer to share a secret x among m participants such that: (1) any unauthorized subset of the participants learns no information about x; and (2) every participant in an authorized subset can perform the computation of a function f on its share to obtain a partial result and these partial results suffice to recover $f(x)$. In a multi-client multi-server setting, HSS can be used to outsource the computation of a function f on the dealer's (clients') private inputs and thus resolve one of the main security issues in outsourcing computation, i.e., the privacy of the client's data. Tsaloli, Liang, and Mitrokotsa (ProvSec 2018) proposed a verifiable HSS (VHSS) model where the partial results of the servers can be verified, in order to resolve another main security issue in outsourcing computation, i.e., the integrity of the outsourced computation. They also constructed a VHSS scheme for computing the product of the dealers' private inputs such that any proper subset of the servers learns no information about the private inputs. In this paper, we present an easy attack of their scheme with which even a single server is able to distinguish between two different sets of private inputs. We propose a new VHSS model and construct a new VHSS scheme for computing the same function. By properly choosing the parameters, our scheme allows cheater detection, cheater identification, robust decoding, and extremely fast verification and result decoding.

1 Introduction

Secret sharing is an important cryptographic primitive and independently introduced by Shamir (1979) and Blakley (1979). An m-party secret sharing scheme allows a dealer to distribute m shares of any secret x to a set $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ of m participants such that: (1) any *authorized subset* $A \subseteq \mathcal{P}$ is able to recover x from their shares; (2) any *unauthorized subset* $B \subseteq \mathcal{P}$ is unable to learn any information about x. The set Γ of all authorized subsets is called an *access structure* and satisfies the *monotone* property: for any $A, B \subseteq \mathcal{P}$, if $A \in \Gamma$ and $A \subseteq B$, then $B \in \Gamma$. The secret sharing schemes of Shamir (1979) and Blakley (1979) realize a *(t, m)-threshold access structure*, which consists of all subsets of $\mathcal{P}$ of cardinality $> t$, and usually called (t, m)-threshold secret sharing schemes. Ito et al. (1987) constructed secret sharing schemes for any general access structures. Since Shamir (1979), Blakley (1979) and Ito et al. (1987), secret sharing has become one of the most important tools in cryptography. For example, they have been extensively applied to construct unconditionally secure multi-party computation protocols (Chaum et al. 1988; Ben-Or et al. 1988), private information retrieval schemes (Chor et al. 1995), threshold cryptographic protocols (Desmedt and Frankel 1989), and attribute-based encryption schemes (Sahai and Waters 2005).

In a homomorphic secret sharing (HSS) (Boyle et al. 2016a), the owner of a secret x can distribute m shares of x, say $s_1, s_2, \ldots, s_m$, to m participants $P_1, P_2, \ldots, P_m$ such that: (1) any unauthorized subset of the participants learns no information about x; (2) each participant P_i of an authorized subset of the participants can perform a computation $\mathsf{Eval}(f, s_j)$ with a function f and its share s_j to get a partial result y^j, and all the partial results suffice to reconstruct the function value $f(x)$. The traditional secret sharing of Shamir (1979), Blakley (1979) and Ito et al. (1987) is a special case of HSS where $f(x) = x$.

✉ Liang Feng Zhang
zhanglf@shanghaitech.edu.cn

[1] School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China

[2] Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China

[3] University of Chinese Academy of Sciences, Beijing 100049, China

HSS has important applications in the area of outsourcing computations (Chen 2016; Xiang and Tang 2015; Yu et al. 2015; Premkamal et al. 2018), which allows resource-restricted users to delegate heavy computations to powerful cloud servers, and in such a way to enjoy the numerous computing resources of the cloud via a pay-per-use manner. In particular, the outsourced computations are usually modeled as the work of computing a function f at an input x. The first schemes (Gennaro et al. 2010; Chung et al. 2010; Benabbas et al. 2011; Parno et al. 2012; Backes et al. 2013; Fiore et al. 2014) for outsourcing computation are in a *single-client single-server* model. Although this model has been prevalent for many years, many different models have been emerging during the past decade. For example, Internet of Things (IoT) (Mattern and Floerkemeier 2010) makes it possible to connect the usual devices, such as sensors, RFIDs, netbooks, and smart phones as an entire network. In a model of outsourcing computation, these devices play the role of the multi-clients and require access to multiple cloud servers, in order to enhance the efficiency, security, and functionality. Such scenarios motivate the emergence of a *multi-client multi-server* model for outsourcing computation. In a multi-client multi-sever model for outsourcing computation, multiple clients can collect data independently and then store their data on multiple servers, and the servers perform the outsourced computations. For example, in an environment with n devices $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, each device $\mathcal{C}_i$ may secret-share a number x_i among m servers $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ such that no proper subset of the servers is able to learn x_i and the computation of a function $f(x_1, x_2, \ldots, x_n)$ can still be offloaded to the servers.

There are two fundamental security problems in outsourcing computation. On one hand, the clients' input (e.g., medical reports) and output (e.g. the result of diagnosis) may be highly sensitive. Disclosure of these information should be strictly prohibited. How to ensure the privacy of the clients' input and output is a fundamental security problem. On the other hand, the outsourced computation is usually heavy and consumes a lot of computing resources. The servers that make profits by providing cloud computing services may have strong financial incentive to run an extremely fast but incorrect computation, in order to free up the valuable computing time for other transactions. How to ensure the correctness of the servers' computation results is also a fundamental security problem. HSS both allows the cloud severs to compute on the outsourced data and ensures the privacy of the clients' inputs. However, a problem that cannot be resolved by HSS (Boyle et al. 2016a) is the integrity of the servers' computations. Motivated by this problem, Tsaloli et al. (2018) introduced the notion of verifiable homomorphic secret sharing (VHSS). VHSS adds verifiability to the traditional HSS, in order to guarantee computation integrity. Tsaloli et al. (2018) proposed two VHSS schemes. In one of their schemes, n clients with private inputs $x_1, x_2, \ldots, x_n$ can outsource the computation of $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$ to m cloud servers. One objective of this scheme is to ensure that any proper subset of the servers learns no information about each private input.

1.1 Our contributions

In this paper, we show that their scheme is not as secure as claimed. In particular, any single server is able to distinguish between two different sets of private inputs. We propose a new simplified model for VHSS and construct a scheme that allows the computation of the same function $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$. In our scheme, there are n clients that provide inputs, m servers and an output client that is responsible to reconstruct $f(x_1, x_2, \ldots, x_n)$. A VHSS scheme is *t-cheater detectable* if the output client is always able to detect the existence of $\leq t$ cheating servers that have provided wrong results. A VHSS scheme is *t-cheater identifiable* if the output client is always able to determine the IDs of $\leq t$ cheating servers that have provided wrong results. A VHSS scheme is *t-robust decodable* if the output client is always able to recover the correct function value, even if $\leq t$ cheating servers have provided wrong results. A VHSS scheme is *outsourceable* if the output client's computation is substantially faster than the naive computation of $f(x_1, x_2, \ldots, x_n)$. Our scheme is t-cheater detectable for all $t \leq (m-1)/2$, t-cheater identifiable and t-robust decodable for all $t \leq (m-1)/3$, and outsourceable whenever $(m-t)\binom{m}{t} \log p + \binom{m}{t} \log^2 p = o(n \log^2 p)$, where p is the size of the finite filed over which the function f is evaluated. In particular, our security/verifiability will depend on no cryptographic assumptions, i.e., our schemes are information-theoretically secure.

1.2 Related work

Homomorphic Secret Sharing. The idea of HSS was introduced by Benaloh (1986). The recent study of HSS was motivated by fully homomorphic encryption (FHE) (Gentry 2009; Rivest et al. 1978) which is a powerful cryptographic primitive and allows the computation of any boolean circuits on encrypted data. The recent study of HSS was initiated by Boyle et al. (2016b), which not only provided a formal definition of HSS but also constructed a 2-party HSS scheme for computing branching programs. In Boyle et al. (2016b), the dealer's data is secret-shared between 2 servers and the data is secret from each individual server, under the DDH assumption. Boyle et al. (2017b) presented several techniques for improving the computational cost of the share conversion procedure in Boyle et al. (2016b). Boyle et al. (2017a) introduced new optimizations that speed-up the implementations of Boyle et al. (2016b) and

Boyle et al. (2017b) by more than a factor of 30 and significantly reduce the share size. Fazio et al. (2017) constructed a HSS scheme based on the circular security of Paillier's encryption and significantly reduced the computational cost of Boyle et al. (2016b) and Boyle et al. (2017b) by reducing the number of required repetitions to achieve a desired overall error round in reconstruction. Dinur et al. (2018) devised a new DDL protocol that further reduced the error probability and thus brought significantly speed-up to the DDH-based HSS schemes. Lai et al. (2018) presented the first plain-model HSS scheme that supports the evaluation of polynomials with degree higher than 2. Boyle et al. (2019) presented new techniques directly yielding efficient 2-party HSS for polynomial-size branching programs from a range of lattice-based encryption schemes, without S/FHE. Compared with the verifiable HSS schemes of this paper, the above schemes have the following differences: (1) In these schemes, the secrecy of the input client's data is computational and based on various cryptographic assumptions such as DDH, the circular security of Paillier's encryption, and learning with errors. (2) In these schemes, the reconstruction of the function value $f(x)$ is not perfect and specifically it will not succeed except with probability $1/\mathrm{poly}(\lambda)$, where λ is the security parameter and $\mathrm{poly}(\lambda)$ is a polynomial function. (3) These schemes never provide verifiability. In particular, if some of the servers are malicious and do not follow the protocols, the output client will output wrong function values.

Multiplicative secret sharing An n-multiplicative secret sharing (MSS) scheme (Barkol et al. 2010) allows a dealer to secret-share n data items $x_1, x_2, \ldots, x_n$ among m participants such that any participant is able to locally convert its shares to a partial result and the sum of the m partial results is equal to $\prod_{i=1}^{n} x_i$. The scheme is said to be t-private if any $\leq t$ out of the m participants cannot learn any information about $x_1, x_2, \ldots, x_n$, even if they have unlimited computing power. The MSS schemes of Barkol et al. (2010) do not provide verifiability. Yoshida and Obana (2017) constructed a verifiably MSS scheme where the participants generate shares that not only allow the reconstruction of $\prod_{i=1}^{n} x_i$ but also allow one to detect the existence of cheating servers. However, their verifications are only done over the reconstructed function values rather than on the shares provided by each individual participant. As a result, they do not provide cheater-identification.

Function secret sharing Function secret sharing (FSS) (Boyle et al. 2015) allows a dealer to secret-share a function f among m participants as $f_1, f_2, \ldots, f_m$, where each function share f_i can be described by a short secret key k_i, such that for any x each participant can compute a partial result $f_i(x)$ and the m partial results sum to $f(x)$, i.e., $f(x) = f_1(x) + f_2(x) + \cdots + f_m(x)$. Such a scheme is said to be t-secure if any $\leq t$ out of the m function shares $f_1, \ldots, f_m$ give no information about f. The original FSS model of Boyle et al. (2015) and the schemes constructed in that model provide no verifiability. Boyle et al. (2016b) constructed a verifiable FSS where an additional m-party interactive protocol is integrated to verify that the keys, generated by a potentially malicious dealer, are consistent with some function f. Quite diffrently, in our VHSS, the verification algorithm is employed to verify if some of the participants are cheating and have provided wrong partial results. In other words, both verifications have completely different purposes.

1.3 Organization

In Sect. 2, we recall both the VHSS model and a construction of Tsaloli et al. (2018), and then present an attack of their construction; In Sect. 3, we propose a new simplified model for VHSS; In Sect. 4, we show a new VHSS scheme that allows the computation of $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$; by adjusting the parameters we obtain schemes that allow t-cheater detection, t-cheater identification, t-robust decoding and outsourceability; Sect. 5 contains our concluding remarks.

2 An attack on Tsaloli–Liang–Mitrokotsa VHSS

In this section, we briefly recall both the VHSS model and the construction of Tsaloli et al. (2018) and provide an attack on its security.

2.1 The model of Tsaloli et al. (2018)

Let λ be any security parameter. We denote by $\mathrm{negl}(\lambda)$ any function that is *negligible* in λ, that is, for any $c > 0$, there is an integer $\lambda_c > 0$ such that $\mathrm{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > \lambda_c$. Tsaloli et al. (2018) proposed a VHSS model for securely outsourcing computations to the cloud. In their model, there are n clients $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ and m servers $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$. For every $i \in [n]$, the client $\mathcal{C}_i$ has a private input x_i. The n clients wish to outsource the computation of a function $f(x_1, x_2, \ldots, x_n)$ to the servers such that no server learns any information about the clients' private inputs and no server can persuade the clients to output a value $\neq f(x_1, x_2, \ldots, x_n)$.

Definition 1 (*VHSS*: Tsaloli et al. 2018) An n-client m-server *verifiable homomorphic secret sharing (VHSS)* scheme for $f : X \to Y$, denoted as $\Pi = (\mathsf{ShareSecret}, \mathsf{PartialEval}, \mathsf{PartialProof}, \mathsf{FinalEval}, \mathsf{FinalProof}, \mathsf{Verify})$, consists of six polynomial-time algorithms as below:

- $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}, \tau_i) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, x_i)$: This is a randomized *secret sharing* algorithm. It takes a security parameter λ, a private input x_i as input and outputs m

Fig. 1 Experiment $\mathsf{Exp}_{\Pi}^{\text{security}}(T, \mathcal{A})$

- The adversary $\mathcal{A}$ gives $(i, x_i, x_i') \leftarrow \mathcal{A}(1^\lambda)$ to the challenger, where $|x_i| = |x_i'|$ and $x_i \neq x_i'$ for all $i \in [n]$;
- The challenger picks a bit $b \in \{0,1\}$ uniformly at random and computes $(\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,m}, \hat{\tau}_i) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, \hat{x}_i)$, where $\hat{\tau}_i$ is an encoded value related to $\hat{x}_i$ and $\hat{x}_i = \begin{cases} x_i, & \text{if } b = 0, \\ x_i', & \text{otherwise;} \end{cases}$
- The adversary outputs a guess $b' \leftarrow \mathcal{A}(\{x_{i,j} : i \in [n], j \in T\}, \{\hat{\tau}_i : i \in [n]\})$, given the shares from the corrupted servers $\mathcal{S}_T$ and the encoded values $\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_n$.

shares $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ and a public verification information τ_i;

- $y^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))$: This is the *partial evaluation* algorithm of the jth server. It takes the n shares $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$ as input, and outputs a partial result y^j;
- $\sigma^j \leftarrow \mathsf{PartialProof}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))$: This is the *partial proof* algorithm of the jth server. It takes the n shares $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$ as input, and outputs a partial proof σ^j;
- $y \leftarrow \mathsf{FinalEval}(y^1, y^2, \ldots, y^m)$: This is the *final evaluation* algorithm. It takes the m partial results $y^1, y^2, \ldots, y^m$ as input, and outputs the final result y;
- $\sigma \leftarrow \mathsf{FinalProof}(\sigma^1, \sigma^2, \ldots, \sigma^m)$: This is the *final proof* algorithm. It takes the m partial proofs $\sigma^1, \sigma^2, \ldots, \sigma^m$ as input, and outputs the final proof σ;
- $\{0,1\} \leftarrow \mathsf{Verify}(\tau_1, \tau_2, \ldots, \tau_n, y, \sigma)$: This is the *verification* algorithm. It takes the public verification information $\tau_1, \tau_2, \ldots, \tau_n$, the final result y, and the final proof σ as input, and outputs either 0 or 1, where 0 means that some servers are cheating and detected, and 1 means that no servers are cheating and y is considered as the correct function value.

A meaningful VHSS scheme should be correct, secure and verifiable (definition omitted from here). The correctness of a VHSS scheme requires that when all algorithms are faithfully executed the final results and the final proofs must both verify successfully and decode to the correct function value.

Definition 2 (*Correctness*) The VHSS scheme Π is said to be *correct* for f if for any n secret inputs $x_1, x_2, \ldots, x_n \in X$, for all $\{(\{x_{i,j}\}_{j=1}^m, \tau_i) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, x_i)\}_{i=1}^n$, all $\{y^j \leftarrow \mathsf{PartialEval}(j, \{x_{i,j}\}_{i=1}^n)\}_{j=1}^m$, all $\{\sigma^j \leftarrow \mathsf{PartialProof}(j, \{x_{i,j}\}_{i=1}^n)\}_{j=1}^m$, all $y \leftarrow \mathsf{FinalEval}(\{y^j\}_{j=1}^m), \sigma \leftarrow \mathsf{FinalProof}(\{\sigma^j\}_{j=1}^m)$, it always holds that $\mathsf{Verify}(\{\tau_i\}_{i=1}^n, y, \sigma) = 1$.

For any set $T \subseteq [m]$, the T-security of a VHSS scheme requires that the servers $\mathcal{S}_T = \{\mathcal{S}_i : i \in T\}$ cannot learn any partial information about the clients' private inputs. Formally, the security is defined with the security experiment of Fig. 1, which is executed between an adversary $\mathcal{A}$ that plays the role of $\mathcal{S}_T$ and the challenger of $\mathcal{A}$. The advantage of $\mathcal{A}$ in guessing b in this experiment is defined as

$$Adv(1^\lambda, \mathcal{A}, T) = \left|\Pr[b' = b] - 1/2\right|, \tag{1}$$

where the probability is taken over the randomness of the challenger and of $\mathcal{A}$.

Definition 3 (*t-Security*) The VHSS scheme Π is said to be t-secure if for any $T \subseteq [m]$ such that $|T| \leq t$, and all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, it holds that $Adv(1^\lambda, \mathcal{A}, T) \leq \text{negl}(\lambda)$.

2.2 The construction of Tsaloli et al. (2018)

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_k$ be a map with the following properties: (1) e is bilinear, i.e., for all $a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; (2) e is non-degenerate, i.e., $e(g, g) \neq 1$; and (3) e is efficiently computable. Tsaloli et al. (2018) proposed a VHSS scheme for computing the product of n group elements $x_1, x_2, \ldots, x_n \in \mathbb{G}$. In their construction, for every $i \in [n]$, the client $\mathcal{C}_i$ has a group element $x_i \in \mathbb{G}$ and an integer $\tilde{x}_i \in \mathbb{Z}_p$ such that $x_i = g^{\tilde{x}_i}$. For any $i \in [n]$, $\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m} \in \mathbb{Z}_p$ are m distinct nonzero field elements; $\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,m} \in \mathbb{Z}_p$ are field elements such that $p_i(0) = \sum_{j=1}^m \lambda_{i,j} p_i(\theta_{i,j})$ for any univariate polynomial $p_i(X)$ of degree $\leq t$. Their scheme can be detailed as follows.

- $\mathsf{ShareSecret}(1^\lambda, i, \tilde{x}_i)$: Pick a polynomial p_i of the form $p_i(X) = \tilde{x}_i + a_1 X + a_2 X^2 + \cdots + a_t X^t$ where $a_i, i \in \{1, 2, \ldots, n\}$ are elements selected uniformly at random, t denotes the degree of the polynomial with $t \cdot n < m$ and $\tilde{x}_i$ is its free coefficient. The algorithm computes $x_{i,j} = g^{\lambda_{i,j} \cdot p_i(\theta_{i,j})}$ for every $j \in [m]$, $\tau_i = e(g, g^{\tilde{x}_i})$, and outputs $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}, \tau_i)$.

- $\mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots, x_{n,j}))$: For the given j and for all $i \in [n]$, multiply all $x_{i,j}$, that is, compute $y^j = \prod_{i=1}^n x_{i,j} = \prod_{i=1}^n g^{\lambda_{i,j} \cdot p_i(\theta_{i,j})}$. Output y^j.
- $\mathsf{PartialProof}(j, (x_{1,j}, x_{2,j}, \ldots, x_{n,j}))$: For the given j and $x_{1,j}, x_{2,j}, \ldots, x_{n,j}$, compute the partial proof, that is, the share of the proof, $\sigma^j = e(g, \prod_{i=1}^n x_{i,j}) = e(g, y^j)$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_k$. Output σ^j.
- $\mathsf{FinalEval}(y^1, y^2, \ldots, y^m)$: Multiply the partial products y^j for $j \in [m]$, that is, compute $y^1 \cdot y^2 \cdots y^m = y$. Output y.
- $\mathsf{FinalProof}(\sigma^1, \sigma^2, \ldots, \sigma^m)$: Multiply the partial proofs to get $\prod_{j=1}^m \sigma^j = \sigma$. Output σ.
- $\mathsf{Verify}(\tau_1, \tau_2, \ldots, \tau_n, \sigma, y)$: Check that $\prod_{i=1}^n \tau_i = \sigma$ and $\prod_{i=1}^n \tau_i = e(g, y)$. Output: 1 if both are satisfied or 0 otherwise.

2.3 Our attack

Tsaloli et al. (2018) claimed that their n-client m-server VHSS scheme for the function $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^n x_i$ is $(m-1)$-secure. That is, for any subset $T \subseteq [m]$ of cardinality $\leq m-1$, the collusion of all servers $\{\mathcal{S}_i : i \in T\}$ cannot learn any information about the group elements $x_1, x_2, \ldots, x_n$. Formally, for any PPT adversary $\mathcal{A}$ that plays the role of the colluding servers $\{\mathcal{S}_i : i \in T\}$, there is a negligible function $\mathrm{negl}(\lambda)$ such that $Adv(1^\lambda, \mathcal{A}, T) \leq \mathrm{negl}(\lambda)$ in the experiment of Fig. 1.

In the scheme of Tsaloli et al. (2018), each client $\mathcal{C}_i$ secret-shares its input $x_i \in \mathbb{G}$ among the m servers with $x_{i,1} = g^{\lambda_{i,1} \cdot p_i(\theta_{i,1})}, x_{i,2} = g^{\lambda_{i,2} \cdot p_i(\theta_{i,2})}, \ldots, x_{i,m} = g^{\lambda_{i,m} \cdot p_i(\theta_{i,m})} \in \mathbb{G}$, where the exponents $p_i(\theta_{i,1}), p_i(\theta_{i,2}), \ldots, p_i(\theta_{i,m}) \in \mathbb{Z}_p$ are the shares of $\tilde{x}_i = \log_g x_i$ under Shamir's t-private threshold secret sharing scheme (Shamir 1979). Furthermore, each client also makes $\tau_i = e(g, g^{\tilde{x}_i})$ public, in order for verification of the servers' results. For every $i \in [n]$, τ_i contains full information about the client's input x_i as $\tau_i = e(g, x_i)$. These group elements may be employed to break the security of the underlying VHSS scheme.

Below we shall show an attack that breaks the system with probability 1. In particular, we show that the VHSS scheme of Tsaloli et al. (2018) is not κ-secure for any integer $\kappa \geq 1$. That is, even one server is able to distinguish between two different vectors of inputs! Without loss of generality, we set $T = [\kappa]$ and consider the set $\{\mathcal{S}_i : i \in [\kappa]\}$ of the colluding servers. It suffices to have a polynomial-time adversary $\mathcal{A}$ such that $\mathbf{Adv}(1^\lambda, \mathcal{A}, T)$ is *non-negligible* in λ. The proposed adversary $\mathcal{A}$ will operate as follows with its challenger:

- *First of all, the adversary $\mathcal{A}$ chooses two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{x}' = (x'_1, x'_2, \ldots, x'_n) \in \mathbb{G}^n$ of group elements such that $x_i \neq x'_i$ and $|x_i| = |x'_i|$ for every $i \in [n]$. In particular, for every $i \in [n]$, the adversary $\mathcal{A}$ knows the discrete logarithms $\tilde{x}_i, \tilde{x}'_i \in \mathbb{Z}_p$ such that $x_i = g^{\tilde{x}_i}$ and $x'_i = g^{\tilde{x}'_i}$. The adversary $\mathcal{A}$ then gives $\tilde{\mathbf{x}} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ and $\tilde{\mathbf{x}}' = (\tilde{x}'_1, \tilde{x}'_2, \ldots, \tilde{x}'_n)$ to its challenger.*
- *The challenger then randomly chooses a bit $b \in \{0, 1\}$. If $b = 0$, it sets $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) = \mathbf{x}$; otherwise, it sets $\hat{\mathbf{x}} = \mathbf{x}'$. The challenger then computes $(\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,m}, \hat{\tau}_i) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, \hat{x}_i)$ for every $i \in [n]$. Then the shares $\{\hat{x}_{i,j} : j \in [\kappa]\}$ and the verification information $\{\hat{\tau}_i : i \in [n]\}$ are given to the adversary $\mathcal{A}$.*
- *Upon receiving these information, the adversary $\mathcal{A}$ will determine if $\hat{\mathbf{x}} = \mathbf{x}$ or $\hat{\mathbf{x}} = \mathbf{x}'$ as follows: if $\hat{\tau}_1 = e(g, x_1)$, then $\mathcal{A}$ outputs $b' = 0$; otherwise, $\mathcal{A}$ outputs $b' = 1$.*

In the above construction, the adversary $\mathcal{A}$ runs in polynomial-time. When $\hat{\mathbf{x}} = \mathbf{x}$, we must have that $\hat{\tau}_1 = e(g, x_1)$; otherwise, we must have that $\hat{\tau}_1 = e(g, x'_1)$. That is, the adversary will always make the correct decision, i.e., $\Pr[b' = b] = 1$. Hence, $\mathbf{Adv}(1^\lambda, T, \mathcal{A}) = 1/2$, which is non-negligible. According to Definition 3, the scheme of Tsaloli et al. (2018) is not κ-secure for any integer $\kappa \geq 1$.

3 Our VHSS model

In this section, we propose a new construction of VHSS scheme in order to securely compute the multiplication of n field elements. In the new construction, we no longer require each input client $\mathcal{C}_i$ to output the public information τ_i for the purpose of verification. In fact, if the τ_i contains information of x_i, then it will potentially be employed by an adversary to successfully distinguish between different input vectors, which has happened in the construction of Tsaloli et al. (2018). Instead, we will follow a model where the algorithms $\mathsf{PartialEval}$, $\mathsf{PartialProof}$ and Verify are removed from Definition 1, the τ_i is removed from the description of $\mathsf{ShareSecret}$, and the algorithm $\mathsf{FinalEval}$ will both verify the correctness of the servers' partial results and output the function value.

Definition 4 (*VHSS*) An n-client m-server *verifiable homomorphic secret sharing* scheme for $f : X \to Y$, denoted as $\Pi = (\mathsf{ShareSecret}, \mathsf{PartialEval}, \mathsf{FinalEval})$, consists of three polynomial-time algorithms as below:

- $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, x_i)$: This is a randomized *secret sharing* algorithm. It takes a security parameter λ, a private input x_i as input and outputs m shares $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ for the servers, one to each server;
- $y^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))$: This is the *jth* server's *partial evaluation* algorithm. It takes the n shares $\{x_{i,j}\}_{i=1}^n$ as input, and outputs a partial result y^j;

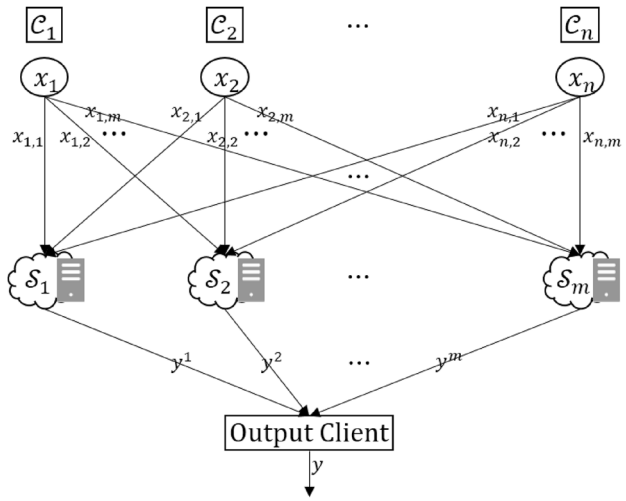

Fig. 2 Our VHSS model

- $\{y, \perp\} \leftarrow \mathsf{FinalEval}(y^1, y^2, \ldots, y^m)$: The *final evaluation* algorithm verifies if the partial results $\{y^j\}_{j=1}^m$ form a valid encoding of the function value $f(x_1, x_2, \ldots, x_n)$. If the encoding is valid, this algorithm outputs the value of $y = f(x_1, x_2, \ldots, x_n)$; otherwise, it outputs $\perp$ to indicate that some of the servers are cheating.

In fact, Tsaloli et al. (2018) has a model (Definition 1 of Tsaloli et al. 2018) for HSS, which provides no verifiability. Our model is identical to theirs except that the final evaluation algorithm FinalEval also performs verification on the servers' partial results, in order to detect the existence of cheating servers.

Our model (see Fig. 2) involves a number of *input clients*, m *servers* $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$, and an *output client*. In this model, any input client with a data item x_i will run the randomized secret sharing algorithm to generate m shares $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ and distribute the m shares to the m servers, one to each server. For every $j \in [m]$, the j-th server will run the partial evaluation algorithm on its shares to produce a partial result y^j. And finally the output client will run the final evaluation algorithm with all partial results, in order to output the function value $y = f(x_1, \ldots, x_n) = \prod_{i=1}^n x_i$.

A VHSS scheme should be correct, secure and verifiable. Intuitively, the correctness of a VHSS scheme requires that the final evaluation algorithm should always accept the partial results that are faithfully computed by PartialEval and output the correct function value.

Definition 5 (*Correctness*) The VHSS scheme Π is said to be correct if for any n secret inputs $x_1, x_2, \ldots, x_n$, for all $\{(x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \leftarrow \mathsf{ShareSecret}(1^\lambda, x_i)\}_{i=1}^n$, for all $\{y^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))\}_{j=1}^m$, it always holds that

$$\Pr[\mathsf{FinalEval}(y^1, y^2, \ldots, y^m) = f(x_1, x_2, \ldots, x_n)] = 1. \quad (2)$$

For any set $T \subseteq [m]$, the T-security of a VHSS scheme requires that the servers $\{\mathcal{S}_i : i \in T\}$ cannot learn any information about the clients' private inputs. Formally, the security will be defined with the security experiment of Fig. 3, which is executed between an adversary $\mathcal{A}$ that plays the role of $\mathcal{S}_T$ and the challenger of $\mathcal{A}$. The advantage of $\mathcal{A}$ in this experiment is

$$Adv(1^\lambda, \mathcal{A}, T) = \left| \Pr[b' = b] - 1/2 \right|, \quad (3)$$

where the probability is taken over the randomness of the challenger and of $\mathcal{A}$.

Definition 6 (*Security*) The VHSS scheme Π is said to be *t-secure* if for any $T \subseteq [m]$ such that $|T| \le t$, and all PPT adversaries $\mathcal{A}$, it holds that $Adv(1^\lambda, \mathcal{A}, T) \le \mathrm{negl}(\lambda)$.

Consider n secret inputs $x_1, x_2, \ldots, x_n$. Let T be the indices of the corrupted servers and $|T| \le m$. Let $\mathcal{A}$ be any PPT adversary that models the corrupted servers $\mathcal{S}_T$. If $\mathcal{A}$ modifies the partial results of the jth server for any $j \in T$, then the modified partial results may cause the clients to both accept the modified partial results and reconstruct a value $\neq f(x_1, x_2, \ldots, x_n)$. The verifiability of the VHSS scheme requires that these servers should succeed with negligible probability. Formally, the verifiability will be defined with the experiment $\mathsf{Exp}_\Pi^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A})$ (see Fig. 4).

Fig. 3 Experiment $\mathsf{Exp}_\Pi^{\mathrm{sec}}(T, \mathcal{A})$

- The adversary $\mathcal{A}$ gives $(i, x_i, x_i') \leftarrow \mathcal{A}(1^\lambda)$ to the challenger, where $|x_i| = |x_i'|$ and $x_i \neq x_i'$ for all $i \in [n]$;
- The challenger picks a bit $b \in \{0, 1\}$ uniformly at random and computes $(\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,m}) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, \hat{x}_i)$, where $\hat{\tau}_i$ is an encoded value related to $\hat{x}_i$ and $\hat{x}_i = \begin{cases} x_i, & \text{if } b = 0, \\ x_i', & \text{otherwise;} \end{cases}$
- The adversary outputs a guess $b' \leftarrow \mathcal{A}(\{x_{i,j} : i \in [n], j \in T\})$, given the shares from the corrupted servers $\mathcal{S}_T$.

Fig. 4 Experiment $\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A})$

- For all $i \in [n]$, generate $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \leftarrow \mathsf{ShareSecret}(1^{\lambda}, i, x_i)$; for every $i \in T$, gives $(x_{1,j}, x_{2,j}, \ldots, x_{n,j})$ to the server $\mathcal{S}_j$;
- The adversary $\mathcal{A}$ outputs a modified multiplicative share $\hat{y}^j$ for every $j \in T$. For all $j \notin T$, the challenger computes $\hat{y}^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))$.
- The challenger then executes $\mathsf{FinalEval}(\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^m)$ to output a value $\hat{y}$. If $\hat{y} \notin \{f(x_1, x_2, \ldots, x_n), \perp\}$, then the experiment outputs 1; otherwise, the experiment outputs 0.

Definition 7 (*Verifiability*) The VHSS scheme Π is said to be *t-verifiable* if for any private inputs $x_1, x_2, \ldots, x_n$, for any $T \subseteq [m]$ such that $|T| \leq t$, and for all PPT adversaries $\mathcal{A}$, it holds that

$$\Pr[\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A}) = 1] \leq \mathrm{negl}(\lambda). \quad (4)$$

Information-theoretic security/verifiability In our definitions of security and verifiability, the adversary $\mathcal{A}$ is allowed be any PPT algorithm. The resulting security and verifiability will be *computational*. If $\mathcal{A}$ is not limited to PPT and the $\mathrm{negl}(\lambda)$ in both definitions is set to 0, then resulting security/verifiability will no longer depend on the assumption that $\mathcal{A}$ runs in PPT, and be called *information-theoretic*. The information-theoretic security/verifiability are strictly stronger than the computational ones. Although our definitions of security/verifiability were given in a computational way in order to be compatible with Tsaloli et al. (2018), our constructions in this paper will have *information-theoretic* security and verifiability. In particular, they require no computational assumptions.

Cheater detection According to Definition 7, in a t-verifiable scheme any $\leq t$ cheating servers will be unable to succeed in the experiment $\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A})$, except with a negligible probability. Whenever the colluding servers fail and the final evaluation algorithm outputs $\perp$, the client will realize that some the servers are cheating. That is, the t-verifiability of a VHSS scheme allows the output client to detect the existence of $\leq t$ cheating servers. We say that such schemes are *t-cheater detectable* or allow *t-cheater detection*.

Cheater tdentification While the property of cheater detection enables us to resolve the computation integrity problem in outsourcing computations, it does not guarantee the identification of cheating servers. This situation may be unsatisfactory because the unidentified cheating servers may always provide wrong results in any future computations and thus prevent the system from properly working. We say that a VHSS scheme is *t-cheater identifiable* or allow *t-cheater identification* if there is an additional algorithm $\mathsf{CheaterID}(y^1, y^2, \ldots, y^m)$ that takes the m partial results as input and in case $\leq t$ out of the m partial results $y^1, y^2, \ldots, y^m$ are incorrect, (i.e., there is a set $T \subseteq [m]$ of cardinality $\leq t$ such that $y^j \neq \mathsf{PartialEval}(j, \{x_{i,j}\}_{i=1}^n)$ for every $j \in T$), the algorithm $\mathsf{CheaterID}(y^1, y^2, \ldots, y^m)$ will output T, the IDs of all cheating servers.

Robust decoding While the property of cheater identification allows the clients to further learn the IDs of cheating servers and reject the partial results of these servers, in many time-sensitive scenarios the clients may still prefer to extract the correct function value $f(x_1, x_2, \ldots, x_n)$ from the remaining (correct) partial results. We say that a VHSS scheme is *t-robust decodable* if there is an additional algorithm $\mathsf{RobustDec}(y^1, y^2, \ldots, y^m)$ that takes the m partial results as input and in case $\leq t$ out of the m partial results $\{y^j\}_{j=1}^m$ are incorrect, the algorithm $\mathsf{RobustDec}(y^1, y^2, \ldots, y^m)$ will output $f(x_1, x_2, \ldots, x_n)$, the correct function value.

Outsourceability When a VHSS scheme is applied to outsource computations, any client that is willing to learn the function value will eventually have to execute the final evaluation algorithm F*inalEval*. We say that a VHSS scheme is *outsourceable* if the output client's computation of running F*inalEval* is significantly faster than the naive computation of $f(x_1, x_2, \ldots, x_n)$. An ideal VHSS scheme should be outsourceable such that the heavy computation of $f(x_1, x_2, \ldots, x_n)$ is offloaded to the servers and the clients are able to learn the function value in an extremely fast way from the servers' partial results.

Robustness of the model We require that the servers do not communicate with each other such that certain subsets of colluding servers cannot learn enough information to determine the data items $x_1, x_2, \ldots, x_n$. The requirement of non-communicating servers is standard in the design of cryptographic protocols such as secret sharing schemes (Shamir 1979; Blakley 1979), secure multiparty computation protocols (Ben-Or et al. 1988; Chaum et al. 1988), multi-server private information retrieval schemes (Chor et al. 1995), distributed oblivious transfer protocols (Naor and Pinkas 2000), and outsourcing computation schemes (Canetti et al. 2012; Ananth et al. 2014). The requirement can be met when the servers belong to different cloud services or have conflicts of interest.

4 Our VHSS scheme

In this section we propose a VHSS scheme under the new model of Sect. 3. Our scheme is information-theoretically secure and verifiable, and allows the computation of the function $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$, where all private inputs $x_1, x_2, \ldots, x_n$ are from a cyclic group $\mathbb{G} = \langle g \rangle$, say the multiplicative group of a finite field. In our construction, there is a parameter t that can be adjusted to achieve the nice properties of cheater detection, cheater identification and robust decoding. In our construction, each private input x_i will be shared among the servers with a CNF secret sharing scheme (Ito et al. 1987). The algorithms of Π are detailed as follows.

- $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, x_i)$: Let $\mathcal{H}_{m,t} = \{H_1, H_2, \ldots, H_M\}$ be the set of all t-subsets of $[m]$, where $M = \binom{m}{t}$. This algorithm chooses M field elements $\{r_{i,\ell}\}_{\ell=1}^{M}$ randomly subject to $\prod_{\ell=1}^{M} r_{i,\ell} = x_i$. For every $j \in [m]$, the share $x_{i,j}$ to server $\mathcal{S}_j$ is defined as

$$x_{i,j} = \{r_{i,\ell} : \ell \in [M], j \notin H_\ell\}. \tag{5}$$

  The algorithm finally outputs m shares $(x_{i,1}, x_{i,2}, \ldots, x_{i,m})$, one to each server.
- $y^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots, x_{n,j}))$: For every $\ell \in [M]$ such that $j \notin H_\ell$, the jth server computes a group element $y_\ell^j = \prod_{i=1}^{n} r_{i,\ell}$. The algorithm then outputs

$$y^j = \{y_\ell^j : \ell \in [M], j \notin H_\ell\}. \tag{6}$$

- $\{y, \perp\} \leftarrow \mathsf{FinaEval}(y^1, y^2, \ldots, y^m)$: For every $\ell \in [M]$, consider the set $\{y_\ell^j : j \in [m] \backslash H_\ell\}$ of partial results. If there exist two different indices $j_1, j_2 \in [m] \backslash H_\ell$ such that $y_\ell^{j_1} \neq y_\ell^{j_2}$, then this algorithm sets $b_\ell = 0$; otherwise, this algorithm sets $b_\ell = 1$ and sets y_ℓ to be the value such that $\{y_\ell^j : j \notin H_\ell\} = \{y_\ell\}$. If $\prod_{\ell=1}^{M} b_\ell = 1$, then this algorithm outputs

$$y = \prod_{\ell=1}^{M} y_\ell; \tag{7}$$

  otherwise, this algorithm outputs $\perp$ to indicate that some servers are cheating.

We shall show that the above scheme is correct, and furthermore for certain values of t the scheme will be t-secure, t-verifiable, t-cheater identifiable or t-robust decodable.

Theorem 1 *The scheme Π is correct.*

Proof When all of the servers are honest, for every $\ell \in [M]$ and every $j \in [m] \backslash H_\ell$, the construction shows that $y_\ell^j = \prod_{i=1}^{n} r_{i,\ell}$. For all $j \in [m] \backslash H_\ell$, y_ℓ^j will be independent of j and equal to each other. Hence, $y_\ell = \prod_{i=1}^{n} r_{i,\ell}$ and $b_\ell = 1$. It follows that $\prod_{\ell=1}^{M} b_\ell = 1$ and the final evaluation algorithm will output

$$y = \prod_{\ell=1}^{M} y_\ell = \prod_{\ell=1}^{M} \prod_{i=1}^{n} r_{i,\ell} = \prod_{i=1}^{n} \prod_{\ell=1}^{M} r_{i,\ell} = \prod_{i=1}^{n} x_i.$$

That is, when the servers are all honest, the final evaluation algorithm will always output the correct function value. □

The t-security of the VHSS scheme Π requires that any t out of the m servers will learn no information about the clients' private inputs.

Theorem 2 *The scheme Π is t-secure for any $t \leq m-1$. In particular, the t-security is information-theoretic such that any t out of the m servers will learn no information about the clients' private inputs, even if they have unlimited computing power.*

Proof It suffices to show that for any subset $T \subseteq [m]$ of cardinality t and any adversary $\mathcal{A}$ that plays the role of the corrupted servers, it holds that $Adv(1^\lambda, \mathcal{A}, T) = 0$. Without loss of generality, we suppose that $T = [t] = \{1, 2, \ldots, t\}$. Then the security experiment will be detailed as follows:

- The adversary $\mathcal{A}$ gives $(i, x_i, x_i') \leftarrow \mathcal{A}(1^\lambda)$ to the challenger, where $|x_i| = |x_i'|$ and $x_i \neq x_i'$ for all $i \in [n]$;
- The challenger picks a random bit $b \in \{0, 1\}$. If $b = 0$, it sets $\hat{x}_i = x_i$ for every $i \in [n]$; otherwise, it sets $\hat{x}_i = x_i'$ for every $i \in [n]$. The challenger then sets $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$. For every $i \in [n]$, it computes $(\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,m}) \leftarrow \mathsf{ShareSecret}(1^\lambda, i, \hat{x}_i)$ as follows:
  - choose M field elements $\{r_{i,\ell}\}_{\ell=1}^{M}$ randomly subject to $\prod_{\ell=1}^{M} r_{i,\ell} = x_i$. For every $j \in [m]$, set $\hat{x}_{i,j} = \{r_{i,\ell} : \ell \in [M], j \notin H_\ell\}$.

    The challenger then gives $\{\hat{x}_{i,j} : i \in [n], j \in [t]\}$ to the adversary $\mathcal{A}$.
- The adversary outputs a guess $b' \leftarrow \mathcal{A}(\{\hat{x}_{i,j} : i \in [n], j \in [t]\})$.

When $t \leq m-1$, it's easy to see that the shares $\{\hat{x}_{i,j} : i \in [n], j \in [t]\}$ are uniformly distributed over the set $\mathbb{G}^{nt\binom{m-1}{t}}$. As a consequence, $\mathcal{A}$ learns absolutely no information about $\hat{\mathbf{x}}$. Hence, we have that $\Pr[b' = b] = 1/2$ and thus $Adv(1^\lambda, \mathcal{A}, [t]) = 0$, which is negligible. Hence, the scheme is t-secure. □

Theorem 3 The scheme Π is t-verifiable for $t \leq (m-1)/2$. In particular, the t-verifiability is information-theoretic such that any t out of the m servers will not be able to persuade the clients to both accept their wrong partial results and

reconstruct a wrong function value, even if they have unlimited computing power.

Proof It suffices to show that for any subset $T \subseteq [m]$ of cardinality t and any adversary $\mathcal{A}$ that plays the role of the corrupted servers, it holds that

$$\Pr[\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A}) = 1] = 0.$$

Without loss of generality, we suppose that $T = [t] = \{1, 2, \ldots, t\}$. Then the security experiment $\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A})$ will be detailed as follows:

- For all $i \in [n]$, the challenger generates $(x_{i,1}, x_{i,2}, \ldots, x_{i,m}) \leftarrow \mathsf{ShareSecret}(1^{\lambda}, i, x_i)$; it then gives $\{(x_{1,j}, x_{2,j}, \ldots, x_{n,j})\}_{j=1}^{t}$ to the adversary $\mathcal{A}$;
- The adversary $\mathcal{A}$ outputs modified multiplicative shares $\hat{y}^j$ for all $j \in T$. For all $j \notin T$, the challenger computes $\hat{y}^j \leftarrow \mathsf{PartialEval}(j, (x_{1,j}, x_{2,j}, \ldots x_{n,j}))$, where $\hat{y}^j = \{\hat{y}^j_{\ell} : \ell \in [M], j \notin H_{\ell}\}$ is a set of field elements;
- The challenger then executes $\mathsf{FinalEval}(\hat{y}^1, \hat{y}^2, \ldots, \hat{y}^m)$ to output a value $\hat{y}$. If $\hat{y} \notin \{f(x_1, x_2, \ldots, x_n), \perp\}$, then the experiment outputs 1; otherwise, the experiment outputs 0.

We show that if the adversary $\mathcal{A}$ ever chooses the value of $\hat{y}^{j^*}_{\ell^*}$ for some $j^* \in T$ and $\ell^* \in [M]$ such that $j^* \notin H_{\ell^*}$ and $\hat{y}^{j^*}_{\ell^*} \neq \prod_{i=1}^{n} r_{i,\ell^*}$, then b_{ℓ^*} will be set to 0. For $\ell^* \in [M]$, a server $\mathcal{S}_j$ receives the shares $\{r_{i,\ell^*} : i \in [n]\}$ if and only if $j \notin H_{\ell^*}$. Then the servers that receive the shares $\{r_{i,\ell^*} : i \in [n]\}$ are exactly $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell^*}\}$ and the the number of such servers is exactly $m - t$. Since $|([m] \backslash H_{\ell^*}) \cap [t]| \leq t$, among the $m - t$ servers at most t are corrupted. Hence, $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell^*}\}$ contains at least $m - 2t$ honest servers. As $t \leq (m-1)/2$, we have that $m - 2t \geq 1$, i.e., at least one of the servers $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell^*}\}$ will be honest. Say the honest server is $\mathcal{S}_{j'}$. Then the honest server $\mathcal{S}_{j'}$ will be able to compute the value $\hat{y}^{j'}_{\ell^*} = \prod_{i=1}^{n} r_{i,\ell^*}$ and includes this field element as its partial results. As there exist two different indices $j^*, j' \in [m] \backslash H_{\ell^*}$ such that $\hat{y}^{j^*}_{\ell^*} \neq \hat{y}^{j'}_{\ell^*}$, the final evaluation algorithm will set $b_{\ell^*} = 0$. As a result, $\prod_{\ell=1}^{M} b_{\ell} = 0$ and the algorithm will output $\perp$. On the other hand, if none of the servers provides wrong partial results, then the final evaluation will accept and output $f(x_1, x_2, \ldots, x_n) = x_1 x_2 \cdots x_n$. That is, whenever $t \leq (m-1)/2$, the final evaluation algorithm's output will always belong to the set $\{f(x_1, x_2, \ldots, x_n), \perp\}$. Hence,

$$\Pr[\mathsf{Exp}_{\Pi}^{\mathrm{verif}}(x_1, x_2, \ldots, x_n, T, \mathcal{A}) = 1] = 0,$$

and so the scheme is t-verifiable. □

Theorem 4 The scheme Π is t-cheater identifiable for $t \leq (m-1)/3$.

Proof It suffices to show that there is an algorithm $\mathsf{CheaterID}(y^1, y^2, \ldots, y^m)$ that can successfully identify all cheating servers when the number of cheating servers is $\leq t$. The description of such an algorithm can be detailed as follows:

- $T \leftarrow \mathsf{CheaterID}(y^1, y^2, \ldots, y^m)$: For every $\ell \in [M]$, consider the set $\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}$ of partial results. Let $y_{\ell} = \textbf{Majority}\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\} \in \mathbb{G}$ be a group element that appears most frequently in the set $\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}$. This algorithm sets $T_{\ell} = \{j : j \in [m] \backslash H_{\ell} \text{ and } y^j_{\ell} \neq y_{\ell}\}$. It finally outputs $T = \cup_{\ell=1}^{M} T_{\ell}$.

For every $\ell \in [M]$, the servers that receive the shares $\{r_{i,\ell} : i \in [n]\}$ are exactly $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell}\}$ and the the number of such servers is exactly $m - t$. Since $|([m] \backslash H_{\ell}) \cap [t]| \leq t$, among the $m - t$ servers at most t are corrupted. Hence, $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell}\}$ contains at least $m - 2t$ honest servers. When $t \leq (m-1)/3$, we have that $(m - 2t) - t = m - 3t \geq 1$, i.e., the majority of the servers $\{\mathcal{S}_j : j \in [m] \backslash H_{\ell}\}$ will be honest. It follows that $y_{\ell} = \textbf{Majority}(\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}) = \prod_{i=1}^{n} r_{i,\ell}$. When $\leq t$ of the servers are cheating, the set T_{ℓ} will consist of the IDs of all cheating servers that wrongly computed y_{ℓ}. Thus, T will be the set of IDs of all cheating servers that wrongly computed at least one of $\{y_{\ell} : \ell \in [M]\}$. □

Theorem 5 *The scheme* Π *is t-robust decodable for* $t \leq (m-1)/3$.

Proof It suffices to show that there is an additional algorithm $\mathsf{RobustDec}(y^1, y^2, \ldots, y^m)$ that can successfully recover the correct function value $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} x_i$, even if $\leq t$ out of the m partial results are incorrect. The description of such an algorithm can be detailed as follows:

- $y \leftarrow \mathsf{RobustDec}(y^1, y^2, \ldots, y^m)$: For every $\ell \in [M]$, consider the set $\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}$ of partial results. Let $y_{\ell} := \textbf{Majority}\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\} \in \mathbb{G}$ be a group element that appears most frequently in $\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}$. This algorithm outputs $y = \prod_{\ell=1}^{M} y_{\ell}$.

Based on the proof of Theorem 4, we have that $y_{\ell} = \textbf{Majority}(\{y^j_{\ell} : j \in [m] \backslash H_{\ell}\}) = \prod_{i=1}^{n} r_{i,\ell}$ and so $\prod_{\ell=1}^{M} y_{\ell} = \prod_{\ell=1}^{M} \prod_{i=1}^{n} r_{i,\ell} = \prod_{i=1}^{n} x_i = f(x_1, x_2, \ldots, x_n)$. Hence, the scheme Π is t-robust decodable for $t \leq (m-1)/3$. □

Theorem 6 *The scheme* Π *is outsourceable as long as* $(m-t)\binom{m}{t} \log p + \binom{m}{t} \log^2 p = o(n \log^2 p)$.

Proof In each execution of the final evaluation algorithm, the client needs to determine M values $\{y_\ell\}_{\ell=1}^{M}$ by looking at the elements of M sets, each with $m-t$ field elements. This step requires $O((m-t)M)$ comparisons of field elements in $\mathbb{F}_p$. The client also needs to multiply the M values $\{y_\ell\}_{\ell=1}^{M}$ to get the function value $f(x_1, x_2, \ldots, x_n)$. This step of computation requires $O(M)$ multiplications modulo p. Each comparison and multiplication require $O(\log p)$ and $O(\log^2 p)$ bit operations, respectively. Therefore, each execution of FinalEval requires $O((m-t)\binom{m}{t}\log p + \binom{m}{t}\log^2 p)$ bit operations. On the other hand, the naive computation of $f(x_1, x_2, \ldots, x_n)$ requires $O(n)$ multiplications modulo p and thus $O(n\log^2 p)$ bit operations. When $(m-t)\binom{m}{t}\log p + \binom{m}{t}\log^2 p = o(n\log^2 p)$, the final evaluation algorithm will be substantially faster than the naive computation and the VHSS scheme will be outsourceable. □

In our VHSS scheme, the numbers m, t and p can be completely independent of n, the number of field elements that will be multiplied. In particular, as our scheme is information-theoretically secure, we are able to choose the m, t and p as small constants, depending on the scenario, such that $(m-t)\binom{m}{t}\log p + \binom{m}{t}\log^2 p = o(n\log^2 p)$ and thus the scheme is outsourceable.

Experimental results We implemented our VHSS scheme with Magma computational algebra system.[1] We executed the programs on a virtual machine that runs an Ubuntu Desktop (Version 16.04.1) operating system, with a 2GB RAM and a 2.3GHz Intel E5-2650 v3 CPU. We chose $p = 101, m = 5, t = 2$ and considered the computation of $\prod_{i=1}^{n} x_i$ for $n = 3000$ randomly chosen field elements $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p$. Our experiments show that the output client's computation (i.e., FinalEval) in our VHSS scheme requires around $t_c = 9 \times 10^4$ CPU clock cycles and the naive computation of the product $\prod_{i=1}^{n} x_i$ requires around $t_n = 10^6$ CPU clock cycles. The experimental results indicate that even for a small number n of data items, the output client's computation in our VHSS scheme is much faster than the naive computation. This fact can also be witnessed by Theorem 6 as $(m-t)\binom{m}{t}\log p + \binom{m}{t}\log^2 p < n\log^2 p$. As the time cost of the output client's computation is independent of n, we expect that the number t_c will be quite steady and the ratio t_n/t_c will keep increasing as n is increasing.

5 Conclusions

In this paper, we proposed an attack on the VHSS scheme of Tsaloli et al. (2018) for computing product function. We simplified the VHSS model of Tsaloli et al. (2018) and provided a new VHSS scheme for computing the same function. Both the security and the verifiability of our constructions are proved in an information-theoretic setting. In particular, we can adjust the parameters m, t, p such that the scheme is t-cheater detectable, t-cheater identifiable, t-robust decodable, and outsourceable.

Acknowledgements The authors would like to thank the anonymous referees for the helpful comments. This work was supported by the National Natural Science Foundation of China under Grant 61602304.

[1] http://magma.maths.usyd.edu.au/magma/.

References

Ananth P, Chandran N, Goyal V, Kanukurthi B, Ostrovsky R (2014) Achieving privacy in verifiable computation with multiple servers—without the and without pre-processing. In: Proceedings of PKC 2014, Springer, Berlin, pp 149–166. https://doi.org/10.1007/978-3-642-54631-0_9

Backes M, Fiore D, Reischuk RM (2013) Verifiable delegation of computation on outsourced data. In: Proceedings of CCS 2013, ACM, New York, NY, pp 863–874. https://doi.org/10.1145/2508859.2516681

Barkol O, Ishai Y, Weinreb E (2010) On d-multiplicative secret sharing. J Cryptol 23(4):580–593. https://doi.org/10.1007/s00145-010-9056-z

Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of STOC 1988, ACM, New York, NY, pp 1–10. https://doi.org/10.1145/62212.62213

Benabbas S, Gennaro R, Vahlis Y (2011) Verifiable delegation of computation over large datasets. In: Proceedings of CRYPTO 2011, Springer, Berlin, pp 111–131. https://doi.org/10.1007/978-3-642-22792-9_7

Benaloh JC (1986) Secret sharing homomorphisms: keeping shares of a secret secret (extended abstract). In: Proceedings of CRYPTO 1986, Springer, Berlin, pp 251–260. https://doi.org/10.1007/3-540-47721-7_19

Blakley GR (1979) Safeguarding cryptographic keys. In: Proceedings of the 1979 international workshop on managing requirements knowledge, IEEE, New York, pp 313–317. https://doi.org/10.1109/MARK.1979.8817296

Boyle E, Gilboa N, Ishai Y (2015) Function secret sharing. In: Proceedings of EUROCRYPT 2015, Springer, Berlin, pp 337–367. https://doi.org/10.1007/978-3-662-46803-6_12

Boyle E, Gilboa N, Ishai Y (2016a) Breaking the circuit size barrier for secure computation under DDH. In: Proceedings of CRYPTO 2016, Springer, Berlin, pp 509–539. https://doi.org/10.1007/978-3-662-53018-4_19

Boyle E, Gilboa N, Ishai Y (2016b) Function secret sharing. In: Proceedings of CCS 2016, ACM, New York, NY, pp 1292–1303. https://doi.org/10.1145/2976749.2978429

Boyle E, Couteau G, Gilboa N, Ishai Y, Orrù M (2017a) Homomorphic secret sharing. In: Proceedings of CCS 2017, ACM, New York, NY, pp 2105–2122. https://doi.org/10.1145/3133956.3134107

Boyle E, Gilboa N, Ishai Y (2017b) Group-based secure computation: Optimizing rounds, communication, and computation. In: Proceedings of EUROCRYPT 2017, Springer, Berlin, pp 163–193. https://doi.org/10.1007/978-3-319-56614-6_6

Boyle E, Kohl L, Scholl P (2019) Homomorphic secret sharing from lattices without FHE. In: Proceedings of EUROCRYPT 2019, Springer, Berlin, pp 3–33. https://doi.org/10.1007/978-3-030-17656-3_1

Canetti R, Riva B, Rothblum GN (2012) Two protocols for delegation of computation. In: Proceedings of ICITS 2012, Springer, Berlin, pp 37–61. https://doi.org/10.1007/978-3-642-32284-6_3

Chaum D, Crépeau C, Damgard I (1988) Multiparty unconditionally secure protocols. In: Proceedings of STOC 1988, ACM, New York, NY, pp 11–19. https://doi.org/10.1145/62212.62214

Chen X (2016) Introduction to secure outsourcing computation. Synt Lect Inf Secur Privacy Trust 8(2):1–93. https://doi.org/10.2200/S00701ED1V01Y201602SPT016

Chor B, Goldreich O, Kushilevitz E, Sudan M (1995) Private information retrieval. In: Proceedings of FOCS 1995, IEEE, Milwaukee, WI, pp 41–50. https://doi.org/10.1109/SFCS.1995.492461

Chung KM, Kalai Y, Vadhan S (2010) Improved delegation of computation using fully homomorphic encryption. In: Proceedings of CRYPTO 2010, Springer, Berlin, pp 483–501. https://doi.org/10.1007/978-3-642-14623-7_26

Desmedt Y, Frankel Y (1989) Threshold cryptosystems. In: Proceedings of CRYPTO 1989, Springer , New York, Berlin, pp 307–315. https://doi.org/10.1007/0-387-34805-0_28

Dinur I, Keller N, Klein O (2018) An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In: Proceedings of CRYPTO 2018, Springer, Berlin, pp 213–242. https://doi.org/10.1007/978-3-319-96878-0_8

Fazio N, Gennaro R, Jafarikhah T, Skeith WE (2017) Homomorphic secret sharing from paillier encryption. In: Proceedings of ProvSec 2017, Springer, Berlin, pp 381–399. https://doi.org/10.1007/978-3-319-68637-0_23

Fiore D, Gennaro R, Pastro V (2014) Efficiently verifiable computation on encrypted data. In: Proceedings of CCS 2014, ACM, New York, NY, pp 844–855. https://doi.org/10.1145/2660267.2660366

Gennaro R, Gentry C, Parno B (2010) Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Proceedings of CRYPTO 2010, Springer, Berlin, pp 465–482. https://doi.org/10.1007/978-3-642-14623-7_25

Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of STOC 2009, ACM, New York, NY, pp 169–178. https://doi.org/10.1145/1536414.1536440

Ito M, Saito A, Nishizeki T (1987) Secret sharing schemes realizing general access structure. IEEE/IEICE global telecommunications conference 1987. Ohmsha Ltd, Tokyo, pp 99–102

Lai RWF, Malavolta G, Schröder D (2018) Homomorphic secret sharing for low degree polynomials. In: Proceedings of ASIACRYPT 2018, Springer, Berlin, pp 279–309. https://doi.org/10.1007/978-3-030-03332-3_11

Mattern F, Floerkemeier C (2010) From the internet of computers to the internet of things. In: From active data management to event-based systems and more. Springer, Berlin, pp 242–259. https://doi.org/10.1007/978-3-642-17226-7_15

Naor M, Pinkas B (2000) Distributed oblivious transfer. In: Proceedings of ASIACRYPT 2000. Springer, Berlin, pp 205–219. https://doi.org/10.1007/3-540-44448-3_16

Parno B, Raykova M, Vaikuntanathan V (2012) How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Proceedings of TCC 2012. Springer, Berlin, pp 422–439. https://doi.org/10.1007/978-3-642-28914-9_24

Premkamal PK, Pasupuleti SK, Alphonse PJA (2018) A new verifiable outsourced ciphertext-policy attribute based encryption for big data privacy and access control in cloud. J Ambient Intell Humaniz Comput 10(7):2693–2707. https://doi.org/10.1007/s12652-018-0967-0

Rivest RL, Adleman L, Dertouzos ML (1978) On data banks and privacy homomorphisms. Found Sec Comput 4:169–179

Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Proceedings of EUROCRYPT 2005, Springer, Berlin, pp 457–473. https://doi.org/10.1007/11426639_27

Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613. https://doi.org/10.1145/359168.359176

Tsaloli G, Liang B, Mitrokotsa A (2018) Verifiable homomorphic secret sharing. In: 12th international conference on provable security. Springer, Berlin, pp 40–55. https://doi.org/10.1007/978-3-030-01446-9_3

Xiang C, Tang C (2015) Efficient outsourcing schemes of modular exponentiations with checkability for untrusted cloud server. J Ambient Intell Humaniz Comput 6(1):131–139. https://doi.org/10.1007/s12652-014-0254-7

Yoshida M, Obana S (2017) Verifiably multiplicative secret sharing. In: Proceedings of ICITS 2017. Springer, Berlin, pp 73–82. https://doi.org/10.1007/978-3-319-72089-0_5

Yu J, Wang X, Gao W (2015) Improvement and applications of secure outsourcing of scientific computations. J Ambient Intell Humaniz Comput 6(6):763–772. https://doi.org/10.1007/s12652-015-0280-0