VerifyNet：安全可验证的联合学习

徐国文，IEEE学生会员，李红伟（通讯作者），IEEE资深会员，

刘森，IEEE学生会员，Kan Yang，IEEE会员，林晓东，IEEE研究员

*摘要*-作为一种新兴的神经网络训练模型，联邦学习因其无需收集用户原始数据即可更新参数而受到广泛关注。然而，由于对手可以从共享梯度中跟踪和获取参与者的隐私，联合学习仍然面临各种安全和隐私威胁。本文主要研究了深度神经网络（DNNs）训练过程中的两个主要问题：（1）如何在训练过程中保护用户隐私（即局部梯度）。（2） 如何验证从服务器返回的聚合结果的完整性（或正确性）。为了解决上述问题，人们提出了几种基于安全或隐私保护的联合学习方法，并将其应用于不同的场景中。然而，在保证用户在培训过程中的隐私的同时，让客户验证云服务器是否正常运行，仍然是一个有待解决的问题。在本文中，我们提出了第一个隐私保护和可验证的联邦学习框架VerifyNet。具体来说，我们首先提出了一个双重掩蔽协议来保证联邦学习过程中用户局部梯度的机密性。然后，云服务器需要向每个用户提供其聚合结果正确性的证明。我们认为，除非能解决模型中的NP难问题，否则对手不可能通过伪造证据来欺骗用户。此外，VerifyNet还支持用户在培训过程中退出。在实际数据上进行的大量实验也证明了该方案的实用性。

*索引术语*-隐私保护，深度学习，可验证的联合学习，云计算。

# 一、 简介

深度学习在许多应用中发挥了重要作用，例如医学预测[？], [?]，自动驾驶仪[？], [?]这种基于深度学习的应用已经渗透到我们社会的方方面面，并逐渐改变了人类在生活、旅行、社交等各个领域的习惯[？], [?].

深度学习需要大量的数据，这些数据通常是从用户那里收集的。然而，用户的数据可能是敏感的或包含一些私人信息。例如，在医疗系统中，患者可能不愿意

|  |
| --- |
| 徐国文在计算机科学与工程学院工作，  中国电子科技大学，成都611731，  中国电科大数据研究院有限公司，550022  中国，贵阳（电子邮箱：徐国文@foxmail.com网站)  李红卫在计算机科学与工程学院，  中国电子科技大学，成都611731，  中国，以及通信安全科学技术实验室，中国成都610041（电子邮件：hongweili@uestc.edu.cn)  刘森现供职于中国电子科技大学计算机科学与工程学院，中国成都611731（电子邮箱：893551724@qq.com)  Kan Yang在美国田纳西州孟菲斯大学计算机科学系工作（电子邮件：干.杨@孟菲斯.edu)  林晓东现供职于盖尔夫大学计算机科学学院，地址：加拿大N12W1，Guelph Stone Rd E 50号（电子邮箱：  xlin08@uoguelph.ca) |

与第三方服务提供商（如云服务器）共享医疗数据[？], [?], [?]. 最近，联合学习[？], [?]它能够在不收集用户原始数据的情况下训练网络，所有用户和云服务器只需共享局部梯度和全局参数就可以协同工作，正逐渐受到学术界和业界的关注。然而，研究表明，攻击者仍然可以间接获取包括tabs[？], [?]和会员资格[？], [?]基于共享梯度。另一方面，联邦学习的数据也经常被报道？], [?]. 在某些非法利益驱动下，特别是在某些非法利益驱动下，云服务提供商可能会返回不正确的结果。例如，一个“懒惰”的云提供商可能会用一个更简单但不太准确的模型压缩原始模型，以降低自己的计算成本，或者更糟的是，恶意伪造发送给用户的聚合结果。因此，保护用户隐私和数据完整性（尤其是服务器返回结果的正确性）是联合学习训练过程中的两个基本问题。因此，设计一个安全的联邦训练协议，在保护用户数据隐私的同时，有效地验证服务器返回结果的正确性，具有重要的现实意义。

为了解决上述问题，人们提出了一些关于隐私保护的深入学习的研究。Shokri等人。[?]提出了一种隐私保护的深度学习协议，通过有选择地共享更新的参数，实现了实用性和安全性的平衡。Trieu Phong等人。[?]结合同态加密和梯度下降技术，提出了一种安全的深度学习系统。最近，Keith Bonawitz等人。[?]利用秘密共享和密钥协商协议，提出了一种实用、安全的联邦学习体系结构，在保证高精度的同时，允许用户在执行过程中离线。但是，上述解决方案都不支持验证从服务器返回的结果的正确性。服务器返回结果的正确性与用户局部梯度的私密性密切相关。一旦对手能够操纵返回给用户的数据，用户隐私被泄露的风险往往会增加。例如，在众所周知的白盒攻击中[？], [?]，对手可以将精心编制的结果返回给用户，用于分析用户上传数据的统计特征，并诱导用户发布更多的敏感信息。

最近有几个计划[？], [?]相继提出了在训练良好的神经网络下缓解数据完整性问题。然而，这些方案要么支持少量的激活功能，要么需要额外的硬件帮助。据我们所知，在训练过程中，没有支持神经网络可验证性的现有解决方案。与一个训练良好的神经网络相比，在训练过程中验证结果的正确性显然更为复杂，因为除了预测结果之外，还需要更新整个网络的参数。此外，如何支持可验证性，同时允许用户在工作流过程中退出（由于不可靠的网络、设备电池问题等），并确保所有用户（包括退出）的局部梯度的机密性也是一个挑战。

在本文中，我们提出了VerifyNet，这是第一个在训练神经网络过程中支持验证的隐私保护方法。我们首先设计了一种基于同态散列函数和伪随机技术的可验证方法，以支持每个用户的可验证性。在此基础上，利用一种不同的秘密共享技术和密钥协商协议来保护用户的局部梯度隐私，并解决了训练过程中的用户退出问题。总之，我们的贡献可以总结如下：

•我们利用与伪随机技术相结合的同态哈希函数作为VerifyNet的底层结构，允许用户以可接受的开销验证服务器返回结果的正确性。

•我们提出了一个双重掩蔽协议来保证联邦学习过程中用户局部梯度的机密性。在训练过程中，它可以承受一定数量的用户因某种原因退出，并且这些退出用户的隐私仍然受到保护。

•我们对我们的VerifyNet进行了全面的安全性分析。我们声称，即使云服务器与多个用户串通，攻击者也不会得到任何有用的用户局部梯度信息。此外，在实际数据上进行的大量实验也证明了我们的VerifyNet是实用的。

本文的其余部分安排如下。在第二节中，我们概述了问题陈述。在第三节和第四节中，我们分别描述了准备工作，并给出了一个技术直觉来解释我们如何解决本文所考虑的挑战。在第五节中，我们描述了VerifyNet的技术细节。然后，在第六节中进行了安全性分析。接下来，分别在第七节和第八节中讨论了性能评估和相关工作。最后，第九节对全文进行总结。

二。问题陈述

在本节中，我们首先回顾联邦深度学习的主要概念。然后描述了系统的体系结构、威胁模型和设计目标。

## A、 深度联合学习

*1） 概述：*根据培训方式，深度学习可分为以下两种类型。

•*集中培训*. 如图1（a）所示，传统的集中式培训从服务器询问用户开始

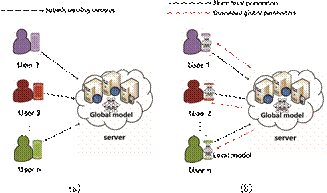


图1：集中培训和联合培训的总体框架。（a） 集中培训。（b） 联合训练。

将他们的本地数据（即训练样本）上传到云端。然后，服务器在云端初始化deep神经网络，用训练样本对其进行训练，直到得到最优参数。最终，云服务器将发布预测服务接口或将最优参数返回给用户。

•*联合训练*. 如前所述，用户直接将本地数据上传到服务器，可能会有隐私泄露的威胁。因此，与集中训练不同，在联邦训练（如图1（b））中，每个用户和服务器协作训练一个统一的神经网络模型。为了加速模型的收敛，每个用户将本地参数（即梯度）共享给云服务器，云服务器聚合所有梯度并将结果返回给每个用户。最终，服务器和每个用户都将获得最佳的网络参数。与集中式训练相比，联合训练降低了用户隐私受到损害的风险。然而，研究表明，攻击者仍然可以基于共享梯度间接获取敏感信息。此外，在某些非法利益的驱使下，恶意云提供商可能会向用户返回错误的结果。因此，本文在验证联邦训练过程中服务器返回结果的正确性的同时，重点保护用户局部梯度的隐私性。

*（二）神经网络：*作为深度学习的底层结构，神经网络可以与各种技术相结合来实现分类、预测和回归。如图2所示，有一个全连接的神经网络，有3个输入，一个隐藏层和2个输出。完全连通是指相邻两层之间的所有神经元通过变量（在本节中称为ω）相互连接。一般来说，神经网络可以表示为函数fx，ωy，其中x表示用户的输入，yˆ是通过带有参数ω的函数f的相应输出。**() = ˆ**

*（三）联合学习更新：*在不损失一般性的前提下，假设每个数据记录是一个观测对⟨x，y⟩，整个训练集为{⟨xi，yi⟩，i=1,2，··T}。*D*

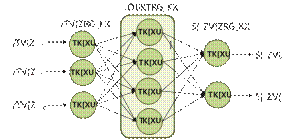


图2：全连接神经网络

损失函数可以在训练集中定义为

（席，益ω）*,*

式中，L（x，yω）=L（y（x，ω）），表示特定的损失函数。本文将损失函数设为（y（x，ω））=l（yyˆ）=| | yyˆ| 2，其中|·| 2是向量的范数。*f,，f我我，f,,我*2

训练神经网络的目的是寻找最优参数ω，从而使损失函数最小化。在我们的验证网中，我们采用随机梯度下降[？], [?]完成这项任务。具体来说，每个参数按如下方式迭代计算。

ω←ω−λ∇L（Dj，ω）*j*+1 *jfj*

式中，ω表示第-次迭代后的参数。是的一个随机子集，是学习率的参数。在我们的联合学习中，每个用户∈N持有一个私有的局部数据集，用一个与所有其他参与者一致的神经网络训练局部集*jjDjDλn公称通径*

前进，其中随机子集=∑⊆N∈N。具体地说，服务器选择第次迭代，然后在*DnDnjj*

用户∈Nj随机选择一个子集⊆Dn执行随机梯度下降。因此，参数更新可以重写如下。*n挪威船级社*



其中=| Dnj |∇L（Dnj，ω）由每个用户计算并随后共享到云服务器。然后，云服务器将全局参数ω返回给所有用户。*ρjnfjj*+1

## B、 系统架构

如图3所示，我们的系统模型由三个实体组成：可信机构（TA）、用户和云服务器。

•*可信机构（TA）*TA的主要工作是初始化整个系统，生成公共参数，并为每个参与者分配公钥和私钥。之后，除非发生争议，否则它将离线。

•*用户*：每个用户需要在每次迭代期间向云服务器发送加密的本地渐变。此外，云服务器还将接收一些其他加密信息，为生成计算结果的证据做准备。

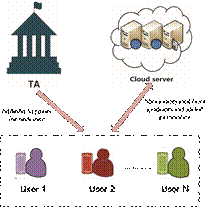


图3：系统架构

•*云服务器*：云服务器聚合所有在线用户上传的渐变，并将结果和证明一起发送给每个用户，我们要求云服务器只知道加密的渐变和最终结果。

## C、 威胁模型与设计目标

在这里我们定义了一个威胁模型，叫做诚实但好奇的安全[？]在我们的验证网里。具体来说，在我们的VerifyNet中，TA是值得信赖的，不会与任何实体串通。包括云服务器在内的所有其他参与者都被认为是诚实而好奇的[？]，这意味着云服务器和用户都将按照约定的协议执行程序，但也可能试图独立推断其他用户的数据隐私[？], [?], [?]. 特别是，我们允许云服务器与多个用户串通获取最具攻击性的能力，也允许云服务器伪造证据（在这里我们不允许共谋伪造证据），并修改计算结果以欺骗用户。

我们的VerifyNet旨在保护用户局部梯度的机密性，同时支持对每个用户的强验证性，并容忍用户在工作流过程中退出。我们声称，除非对手能够解决我们模型中采用的NP难问题，否则欺骗是不可能成功的。

三、 准备工作

为了便于理解本文，我们介绍了VerifyNet中使用的一些密码原语，这将使读者更容易理解我们的方法。

## A、 双线性配对

双线性对可以表示为一个映射：×G2→GI，其中和都是素数阶相同的乘法循环群。在不失一般性的前提下，我们假设和的生成元分别是和。非正式地说，双线性对具有以下性质。*eG*1 *G*1 *G*2 *问G*1 *G*2 *ghe*

1） 双线性性：给定随机数∈Zq＊，对于任意∈G1和∈G2*a、 bg*1 *g*2 ，我们有*e*（g12）。*，克ab型*

2） 可计算性：（g1，g2）可以有效地计算任意∈g1和∈g2。*eg*1 *g*2

3） 非简并：（g，h）=1̸，其中和分别是和的生成元。*eghG*1 *G*2

## B、 同态散列函数

非正式地，给定一个消息∈Zq，一个抗冲突同态散列函数[？], [?]：Zq→G1×G2可指示如下。*席氢氟酸*

*氢氟酸*（x） =（A）=（g（x）（x））*我我，B我氢氟酸δ,ρ我，小时氢氟酸δ,ρ我*

其中和都是在有限域中随机选择的密钥。（）是单向同态散列函数。x1（更精确地说）=*δρZq公司HFδ，ρ氢氟酸*

（g（x1）（x1）），（x2）=（g（x2）（x2）），同态散列函数具有以下性质。*氢氟酸δ,ρ，小时氢氟酸δ,ρ氢氟酸氢氟酸δ,ρ，小时氢氟酸δ,ρ*

1） 可加性（在指数中）可以表示为（x1+x2）←（g（x1）+HF（x2），*氢氟酸氢氟酸δ,ρδ,ρ*

*h氢氟酸δ,ρ*（x1）+HF（x2））*δ,ρ*

2） 乘以常数可以表示为（αx1←（g（x1），（x1））。*α氢氟酸*)*αHFδ,ρhαHFδ,ρ*

同态散列函数还有其他有趣的性质。感兴趣的读者可以参考[？], [?]更多细节。

## C、 伪随机函数

我们采用了Dario Fiore等人设计的伪随机函数。[?]在我们的验证网里。非正式地说，给定密钥=（K1，K2），伪随机函数：{0,1}∗×{0,1}\*→G1×G2由另外两个伪随机函数组成，即：{0,1}\*→Zq2和：{0,1}\*→Zq2。给定一个输入（I1，I2），我们有（I1）=*KPFK公司PFK公司*1 *PFK公司*2 *PFK公司*1

（γI）和（I2）=（γI）。因此，我们1*，νI*1*PFK公司*22*，νI*2

*PF公司K*（I12）=（E，F）=（g12+ν1212+ν12）*，我γ我γ我我ν我，小时γ我γ我我ν我*

作为验证的一部分，将利用伪随机函数来验证来自云服务器的结果的正确性。

## D、 秘密共享协议

在VerifyNet中，我们使用Shamir&apos;s-out-of-N秘密共享协议[？]将秘密分成N个独立的部分，其中N表示模型中的用户数，并且是阈值。这意味着任何大于的共享子集都可以用于恢复机密。具体地说，实现这个秘密共享协议涉及以下步骤。*tstts*

1） S.share（S，t，U）→{（n，sn）}∈U：在给定阈值≤| U |和秘密的情况下，输出每个用户的共享，其中U表示在有限域F中指定的一组用户ID（假定是唯一的），且| U |=n。*nts序号sn*

2） S.recon（{（n，sn）}∈M）→S：输入一个子集M，其中∈M⊆U且≤| M，输出秘密。*n，吨nts*

## E、 关键协议

Diffie Hellman密钥协议[？], [?]在我们的VerifyNet中也被采用，为任意两个用户创建共享密钥。具体地说，给定一个具有素数顺序的组，每个用户的密钥/公钥被创建为卡根（G，G，q）→（SKn），其中是群的生成元。密钥和密钥分别是公开的和公开的。然后，给定用户的公钥，用户和用户之间的共享密钥可以生成为同意（SKn，gSK）→sn，m。在实际应用中，为了方便起见，通常设置为（（gSK）SK）。*G问nn，葛兰素史克ngGSKn公司葛兰素史克葛兰素史克米米n米米米n序号，mH*

# 四、 技术直觉

如上所述，在联合学习中，每个用户需要将其本地梯度提交给云，然后从服务器接收聚合结果（所有本地梯度的总和）。然而，有三个问题需要解决。首先，我们需要保护用户的局部梯度的隐私，因为对手可以通过这些梯度信息间接地破坏用户的敏感信息。其次，为了防止服务器恶意欺骗，每个用户都应该能够有效地验证服务器返回结果的正确性。第三，在实际场景中，由于不可靠的网络或设备电池问题，用户无法按时将数据上载到服务器是非常常见的。因此，我们提出的协议应该支持用户在训练过程中由于某种原因离线。在本节中，我们给出了一个技术直觉来解释我们如何解决这三个挑战。

## A、 单屏蔽保护用户的梯度

假设每个用户都有一个局部梯度（n∈U，| U |=n），我们原本打算设计一个单一的掩蔽协议来保护用户梯度的隐私。具体来说，假设我们系统中所有用户的ID都是有序的，并且任意两个用户都同意一个随机数。然后，我们可以加密每个用户的局部梯度，如下所示。*nxn公司，n米雷诺数，米nxn公司*

（一）

因此，在每个用户向服务器提交其加密的梯度∑xˆn后，它可以计算聚合梯度如下。∈U*xn公司n*

z=∑xˆ=∑x（2）∈U∈U*nnnn*

然而，这种方法有三个缺点。首先，每个用户需要与所有其他用户协商一个随机数，这将导致二次通信开销（U2）。第二，离线协议在用户培训过程中的失败。我们注意到，即使只有一个用户没有按时上传数据，由于无法取消该用户梯度中添加的随机数，上述聚合操作也无法成功完成。第三，上述协议不支持可验证性。服务器返回结果的正确性与用户局部梯度的私密性密切相关。一旦对手能够操纵数据的完整性，用户隐私被泄露的风险往往会增加。因此，安全协议还应支持服务器返回结果的可验证性。*雷诺数，米O*

## B、 支持可验证性的双掩蔽Potocol

针对单掩蔽协议存在的问题，提出了一种双掩蔽协议。我们首先利用伪随机发生器[？]和Diffie Hellman密钥协议[？], [?]在两个用户和之间生成随机数。具体来说，我们首先要求TA为每个用户随机创建密钥对（NnPK，NnSK）。然后，我们要求云服务器向所有用户广播所有公钥∈U。最后，利用伪随机发生器和Diffie-Hellman密钥协商，每两个用户都可以生成一个约定的随机数，表示为←卡。同意（NnSK，）。因此，每个用户的局部梯度可以加密如下。*雷诺数，米n米nNnPK键，nn米序号，mNmPK公司xn号*

                                *十*ˆ*n*=x*n*+ ∑PRGPRG公司（第*n、 米*)（三）

*米*∈U:n<m

其中PRG（sn，m）是带种子的伪随机发生器。*sn、 米*

接下来，我们采用门限秘密共享方案[？]在培训过程中支持用户离线。简言之，为了抵消掉站用户梯度中增加的随机数，每个用户利用门限秘密共享方案提前将其密钥共享给其他所有用户。因此，如果一个用户不能按时向云端提交数据，服务器可以通过要求超过阈值的用户提交用户的秘密份额来解密所有其他用户的m（m≠n∈U）中的随机数PRG（sn，m））。这样，随机数PRG（sn，m）可以被恢复并最终从ˆm中移除，但是仍然存在一个问题。在某些时候，一些用户可能会延迟将数据上传到云端，这可能会导致服务器错误地确定这些用户处于脱机状态，并要求其他在线用户上载这些用户的共享以删除随机数。然而，就在这时，这些用户成功地将他们的ˆn上传到了云端。因此，由于服务器拥有足够的这些用户的秘密共享，它可以通过移除所有随机数来获得。为了解决这一问题，我们在每一个随机噪声呼叫中增加一个新的随机噪声呼叫，并利用门限秘密共享方案将其共享给所有用户。因此，每个用户的局部梯度加密如下。*n北朝鲜n十十n十十xn公司序号，mβn十βnxn公司*

*十*ˆn=xn+PRG（βn）+∑PRGPRG（sn，m）

*米*∈U:n<m

（四）

这样，一旦需要解密以获得聚合结果∑xn，云服务器只能接收所有在线用户的共享，以及所有退出用户的共享，因为这些信息足以进行解密操作。*βn北朝鲜*

双屏蔽协议主要是为了保护用户在培训过程中的数据隐私，支持用户在培训过程中因故离线。但是，它缺乏对可验证性的考虑，即没有设计具体的可验证机制。因此，双掩蔽协议不支持验证从服务器返回的聚合结果的正确性。我们希望设计一个可验证的解决方案，它与我们的双重屏蔽协议高度兼容，并且允许每个用户在没有可信的第三方参与的情况下轻松验证从服务器返回的结果的正确性。为了应对这一挑战，我们将同态散列函数与伪随机技术相结合，作为我们可验证方法的底层结构，允许用户以可接受的开销验证云服务器执行的正确性。具体验证过程见第五节。

五、 拟定方案

在本节中，我们将介绍技术细节。从较高的层次来看，VerifyNet的目的是解决联邦训练过程中存在的三个问题。一是保护用户在工作流中的局部渐变的隐私。其次，为了防止服务器恶意欺骗，我们的VerifyNet支持每个用户有效地验证服务器返回结果的正确性。第三，VerifyNet在培训过程中也支持离线用户。

图4显示了我们的VerifyNet的详细描述，它包括五个回合来完成上述任务。具体来说，TA首先初始化整个系统并生成VerifyNet中所需的所有公钥和私钥。然后，每个用户加密其本地渐变并将其提交给云服务器。在收到来自所有在线用户的足够消息后，云服务器聚合所有在线用户的梯度，并将结果连同结果一起返回给每个用户。最后，每个用户决定通过验证来接受或拒绝计算结果，并返回到第0轮开始新的迭代。*nxn公司证据证据*

如第四轮所示，具体核查过程如下。

*正确性证明：*

*n*=| U3|n=| U3|

（A，B）=（A∏B）=1 n=1*n,     nn*

=（∑∈U3（x）∑∈U3（x））=（（∑∈U3）（∑∈U3））*gn氢氟酸δ,ρn，小时n氢氟酸δ,ρng氢氟酸δ,ρn十n，小时氢氟酸δ,ρn十n*

=（A′，B′）

*e*（A，h）=e（g（σ））=e（g，h（σ））*氢氟酸δ,ρ，小时氢氟酸δ,ρ*

=e（g，B）e（L，h）=e（g∑∈U3+ν−HF（x））1/d（5）*nγnγnνδ,ρn，小时*

∑1∈1/d*g、 hnγnγnνδ,ρn*)

=e（g，Q）e（A，h）·e（L，h）d

=e（g（σ））·e（g∑∈U3+ν−HF（x））*氢氟酸δ,ρ，小时nγnγnνδ,ρn，小时*

=e（g，h）∑∈U3+ν*nγnγnν*

= Φ

如果以上公式中的任何一个无效，则拒绝聚合结果。否则，接受结果并进入第0轮。用户和云服务器迭代运行0-4轮，直到整个神经网络配置满足预先设置的约束。

六、 安全性分析

在本节中，我们首先简要描述验证的正确性。然后，我们分析我们的VerifyNet如何保证每个用户的局部梯度的机密性。其他安全指标超出了本文的范围。

|  |  |
| --- | --- |
|  | VerifyNet的实现过程 |
| • | 初始化0（舍入）：  *助教*:  -    （pnk1，和pnk2）分别是私钥（p2/K1），其中∈（K2/n）是私钥，其中∈n是私钥。*nK*  （NnPK，NnSK），（PnPK，PnSK）将被用来加密用户的局部梯度。*nxn公司*  *用户n*:  -    通过安全通道将公钥（NnPK，PnPK）发送到云服务器。  *服务器端*:  -    从至少个用户（表示为U1⊆U）接收消息，其中是我们模型中使用的Shamir&apos;s-out-of-N协议的阈值。否则，中止并重新开始。*ttt* |
| • | -    广播{m，NmPK，PmPK，τ=sum}∈U1给每个用户∈U1，其中=sum表示要计算的统计标签。*米τ*  第1轮（密钥共享）：*用户n*  -    从云服务器接收{m，NmPK，PmPK，τ=sum}∈U1。检查| U1 |≥t，所有密钥对（NmPK，PmPK）是否不同。否则，中止并重新开始。*米*  -    选择一个随机数。生成as{（m，βn，m）}∈U1←S.share（βn，t，U1）的份额，其中是用户对用户的份额*米*. *βnβn米βn，mn*  -    生成P←N（as{）（.），P}∈U1），N←||共享女士||N（N | |）U1∈U），其中1，恩西斯在哪里user（）的共享表示对用户的对称加密。[?]与*nSKnPK公司，P米PK键m、 Nn、 mSKnSK公司mmPK公司n、 mSK公司nSK公司βn、 米，t，米Nn、 mSK公司n米*  -    计算航空发动机卡。同意（P密钥卡。同意（P*n、 米*  -    发送{P}∈U1到云服务器。*n、 米米*  *服务器端*: |
| • | -      接收来自至少个用户的消息（表示为U2⊆U1）。否则，中止并重新开始。-广播{P}∈U2给每个用户∈U2。*tm、 n米*  第二轮（屏蔽输入）：*用户n*  -      接收{P*m、 n*}*米*∈U2 从云服务器。检查是否∈U←U2 ⊆U1*nSK公司*和, |美国*N*2*英里/公里*| ≥).*t*. 否则，中止并重新开始。-计算每个用户的共享密钥*米*2 作为*序号，m*卡。同意（N）  -      将局部梯度加密为ˆn=xn+PRG（βn）+PRGPRG（sm，n）。*十*  *米*∈U∑2:n<m  -      为了验证将来从服务器返回的结果的正确性，一些附加信息的计算如下。  计算（x）=（A）=（g（x）（x））。*氢氟酸nn，Bn氢氟酸δ,ρn，小时氢氟酸δ,ρn*  计算（n）=（γn，νn）；PFK2（τ）=（γ，ν）。*PFK公司*1  计算（n，τ）=（E）=（g+ν+ν）*PF公司Kn，Fnγnγnν，小时γnγnν*  -      计算=（En·A−）1/d=（gγnγ+νnν-）1/d，其中是选定的正整数。*自然对数n*1*HFδ，ρ*（新）*d*  -      计算=（Fn·Bn−）1/d=（hγnγ+νnν-）1/d，其中是选定的正整数。*问题*1*ρHF，δ*（新）*d*  -      发送=（xˆn，An，Bn，Ln，Qn，Ωn=1）到云服务器。*σn*  *服务器端*: |
| • | -    接收至少3个用户的消息（表示为∈U2U3⊆U2）。否则，中止并重新开始。广播对每个用户*t*.  -  第三轮（揭开面纱）：*用户n*  -    解密每个P*n、 米*  -    inCheck whetherRound1，但是在上传数据到服务器之前退出，在）| Um3∈U⊆U，m22∈U\Uand32}\{Uand3}|≥{as（βtn。如果不是，则中止并重新开始。| | m）| | mN∈U | | 3}到云，其中←声发射(KA.同意2. U2\U（3表示已向服务器发送数据的用户），P）。*n、 mSK公司nn、 米n、 mSK公司βn、 米PnSK公司，P英里/公里n、 米*  -    发送{（N  *服务器端*:  -    接收至少个用户的消息（表示为U4⊆U3）。否则，中止并重新开始。*t*  -    计算←S.recon3（{Nn，mSK}m4）4（，t）。2 3 3）条）*北朝鲜*∈U  -    计算←S.recon（{β（}∈U）。}∈U\U，m∈U*βnn、 米米，吨SK公司，N英里/公里n*.  --计算PRGPRG（（sβn，mn）n∈U）←.PRG卡。同意{N*n*  -    ∑计算所有用户的聚合渐变3 *十n*= ∑*n*∈U3 *十*ˆ*n*−∑*n*∈U3 珠江三角洲∈U3 作为*，米*珠江三角洲（第*n、 米*) +*n*∈U3*，米*∈U∑2\美国3：n>米珠江三角洲（第*m、 n*).  *n*∈U  -- 计算*n*|美国|*证据*{A，B，L，Q，n|美国Ω}聚合坡度如下。*n*|美国|*n*|美国|*n*|美国. |
| • | -    向每个用户∈U4广播{σ=∑∈U3Ω}。*克雷索nxn，A，B，L，Q，*  第四轮（核查）：*用户n*  -    已知（？）？n） =（γn，νn）和（τ）=（γ，ν），计算=？∑∈U3（·γnγ+νd）和Φ=e（g，h）φ。*PF公司K*1*PF公司*?*K*2?*φnn*.*ν*  -    验证（A，B）=（A′），e（A，h）=e（g，B）；（L，h）=e（g，Q），Φ=e（A，h）e（L，h）*，Be*  -    如果以上公式中的任何一个无效，则拒绝聚合结果。否则，接受结果并进入第0轮。 |

图4:VerifyNet的详细说明

## A、 验证的正确性

如第五节所示，在从云服务器接收到{σ，A，B，L，Q，Ω}后，每个用户首先检查Φ=e（A，h）·e（L，h）d？]，Φ=e（A，h）·e（L，h）d仅当∑∈U3+ν包含在（的指数中）时才成立。如果是这样，每个用户都可以推断*我BDHI公司nγnγnν我g*

*我*==π| U3 |=g∑∈U3+ν−HF（σ），知道=1*n我nnγnγnνδ,ρn*

*A*=gHF。之后，根据假设[？]，每个用户进一步检查（A，h）=e（g，B）和（L，h）=e（g，Q）是否保持。如果这是真的，每个用户都会相信云服务器正确计算和。在此之前，每个用户都已验证（A，B）的计算是否正确。最后，如果（σ）=（A′，B′=（A，B′）=（A，B）成立，每个用户都相信云服务器确实返回了正确的聚合结果=∑3。*δ,ρ*(σ)*DDH公司eeB问氢氟酸σn*∈U*xn公司*

这里我们省略了详细的证明，因为利用-[？]和假设[？].*我BDHI公司DDH公司*

## B、 诚实但好奇的安全感

在这一节中，我们证明了我们的VerifyNet在诚实但奇怪的环境下是安全的。在我们的威胁模型中，云服务器可以与任何-1个用户串通以获得最具攻击性的能力，但是他们仍然对诚实用户的本地梯度一无所知，除了聚集的结果。如上所示，每个用户的局部梯度被加密为*txn公司*

*十*ˆn=xn+PRG（βn）+∑PRG（sn，m）

*米*∈U2:n<m

−PRG（sm，n）

*米*∈U2:n>m

此外，每一个也用于同态散列函数[？], [?]生成部分验证信息。因为同态散列函数已经被证明是安全的[？]，这里我们主要讨论ˆn可以达到的隐私保护水平。在正式呈现完整的证明过程之前，我们先介绍一些稍后将使用的有用符号。*xn公司十*

我们知道用户可能会在某个工作时间退出。我们使用U⊆U来表示在第1轮时将数据顺利上传到云服务器的用户。因此，我们有U⊇U1⊇U2⊇U3⊇U4。利用符号U\ui来表示在第−1轮中已向服务器发送数据，但在第1轮上载数据之前退出的用户。如前所述，假设每个用户都有一个局部梯度（n∈U），我们采用U′={xn}∈U′表示局部梯度的子集U′，其中U′⊆U。*我我我我*+1 *我我nxn公司，十n*

在我们的VerifyNet中，一方的视图被定义为其内部状态（包含其输入和随机性）以及从其他方接收的所有消息。需要注意的是，当一方退出执行时，该方将立即停止接收消息。

为了简单起见，我们使用表示云服务器。给定一个子集W⊆U∪{S}，W中各方的共同观点可以表示为一个随机变量（xu1u2u3u4），其中和表示*S***真实的**U、 t，k*,,,,tk*

W

在我们的协议中分别使用了阈值和安全参数。

接下来我们将给出两个定理。第一个定理表明，任何合谋行为（不包括云服务器）小于用户在获取其他用户私有信息时除了聚合的结果。*t*

定理1（防御来自多重攻击的联合攻击

用户）。U3⊆U2⊆U1⊆UkW⊆UU | U |≥tU4⊆*对所有人t，, , 十, 具有SIM卡，谁的*

## ，有一个PPT模拟器

*输出与的输出无法区分***真实的**U、 t，k*.*

W

**真实的**

≡

**SIM卡**美国U2U4（U3瓦）*，t，k,,,*

*证据。*因为我们排除了云服务器的参与，所以在set W中各方的联合视图并不依赖于其他非W用户的输入，因此模拟器可以通过使用诚实但好奇的用户的真实输入运行协议来生成一个完美的模拟，但是用假数据（比如随机生成的数字）代替诚实用户的输入。我们强调W中用户的模拟视图与真实视图的输出是不可区分的。更具体地说，在第2轮中，模拟器通过使用随机数（如0）而不是使用真梯度为所有诚实用户（不在W中）生成屏蔽输入ˆn。另外，我们注意到服务器只是在这一轮的解密过程中发送一个所有在线用户的ID列表，而不是具体的ˆn的实际值，这意味着诚实而好奇的用户无法识别云服务器返回的计算结果是否基于诚实用户的真实梯度。因此，W中用户的模拟视图与真实视图的输出是无法区分的。*十十***真实的**U、 t，k

W

定理2（防御来自云服务器和多个用户的联合攻击）。对于所有UW⊆U∪{S}| W\{S}|<tU | U |≥tU4⊆U3⊆*t，k，十, , , 具有，和*

U2⊆U1⊆U*，有一个PPT模拟器SIM，其输出与***真实的**U、 t，k*.*

W

**真实的**美国W（徐，U1U2U3U4）*，t，k,,,*

≈

**SIM卡**

## 哪里

∑xif | U3 |≥tξ=∈U3\W*nn*

                                             ⊥否则

*证据。*我们用一个标准的混合参数来证明我们的定理2。其主要思想是模拟器SIM对我们的协议进行了一系列的修改，最终使模拟视图与真实视图无法区分。在我们的混合参数中，W{i=1，···9}表示对W原始协议的安全修改，这确保修改后的操作与原始操作不可区分。**西姆雷亚比**U、 t，kU、 t，k*,*

**hyb1型**在这个混合体中，模拟器改变所有诚实用户的行为，其中∈{U2\W}。具体来说，为了*nn*

每个用户，选择一个统一的随机数来代替共享密钥卡。同意（PnSK，PmPK）用户与在同一集中，并执行加解密功能。例如，每一个诚实的用户都使用第1轮生成密文，而不是利用卡。同意（PnSK、PmPK）。DDH假设[？]确保这种混合体具有与实际协议的不可区分性。*nvn，米n米vn，米n、 米***hyb6型**

**hyb2 hyb7 hyb3**在这个混合体中，模拟器用随机值的加密共享（例如，0，具有适当长度）替换诚实用户（在集合{U2\W}）发送给其他用户的所有加密数据（即和的加密共享）。然而，所有诚实的用户在这一轮的解密过程中仍然将正确的共享返回给云服务器。因为我们只是改变了密文的内容，对称认证加密的性质[？], [?]确保这个混合体与真实协议之间的不可区分性。在这个混合体中，我们首先定义一个子集如下。*βn北朝鲜*

                                                               \如果ξ=⊥

否则

然后，在密钥共享的回合中，对于集合U\*中的所有诚实用户，模拟器将的所有共享替换为随机值（例如，0，具有适当长度）。很明显，对手不会得到额外的份额，或者因为诚实的用户不会透露他们的份额。|U3 |≥t，U\*=U2\U3\W），或者因为所有诚实的用户都离线（resp。|U3 |<t，U∗=U2\W，其中=⊥）。Shamir的秘密共享方案的安全性保证了即使拥有当前秘密的−1份也不可能恢复秘密，这意味着即使是诚实而好奇的用户也拥有| W |<t个共享，他们仍然无法分辨诚实用户提交的共享是否来自真实。*nβnβnβnξtβnβn*

**hyb4和hyb8**In this hybrid, instead of generating PRG(βn) for all users in the set U∗, the simulator uses uniformly random number with appropriate size to replace it. It is easy to understand that the simulator just substitutes the output of PRG, where PRG is the Pseudorandom Generator [?] mentioned before. Therefore, the security of Pseudorandom Generator [?] ensures that this hybrid is indistinguishable from the real protocol.*rn*

**hyb5 hyb9** In this hybrid, for each user in the set U∗, the simulator generates the masked input as below:*n*

*x*ˆn = rn + ∑ PRGPRG(sm,n)

*m*∈U2:n<m

instead of utilizing

*x*ˆn = xn+PRG(βn) + ∑ PRG(sn,m)

*m*∈U2:n<m PRG(sm,n)

is also a random value, and it is easy to deduce that the distribution of and +PRG(βn) is indistinguishable. It should be noted that if =⊥, the simulator has already completed the simulation (describe as ) since SIM successfully simulates REAL without knowing for all parties ∈ W. Hence for all the simulations that follow, we assume ≠ ⊥.*rn xnξ* **hyb5***xn n ξ*

In this hybrid, for every user ∈ U3\W, the simulator substitutes the shares of with shares of random values (e.g., 0, with appropriate length). Similar to , the security of Shamir&apos;s secret sharing protocol guarantees that this hybrid is indistinguishable from the real protocol. In this hybrid, given a user ∈ U3\W, for all other users ∈ U3\W, the simulator uniformly selects a random number to replace the sharked key (i.e., = KA.agree{NnSK,NmPK}) between user and , and this random number will be used as the seed of PRG for both user and .*n NnSK* **hyb3***m*′ *n sn,m*′ *n m*′*n m*

Specifically, a random value is selected for each user ∈ U3\W\{m′}. Instead of sending*s*′*n,m*′ *n*

*x*ˆn = xn+PRG(βn) + ∑ PRG(sn,m)

*m*∈U2:n<m

2:

SIM submits

*x*ˆ=x+ r+ ∑PRG(s)*n n n m n,m*

*m*∈U2\{ }:n<m′

PRG(s) + ∆′ · PRG(s′n,m′)*m,nn,m*

where ∆n,m= 1 if . Otherwise, ∆n,m= −1.′ *n < m*′′

Correspondingly, we have

PRG

Similarly, the DDH assumption [?] ensures that this hybrid possesses the indistinguishability from real protocol. In this hybrid, for the same user selected in previous hybrid and all other user ∈ U3\W, the simulator also uniformly selects a random number ′ to replace the computation of PRG(s′n,m). Similar to , it is easy to understand that the simulator just substitutes the output of PRG. Therefore, the security of Pseudorandom Generator [?] ensures that this hybrid is indistinguishable from the real protocol.*m*′ *n rn,m*′**hyb4**

In this hybrid, for each user in the set U3\W, the simulator submits*n*



instead of sending

*x*ˆn =xn + PRG(βn) + ∑ PRG(sn,m)

*m*∈U2:n<m

PRG(sm,n)

where *Rn n*

Since PRG(βn) has been changed in the previous hybrid { } ∈Uis random value selected by the simulator,and subjected to3\W *Rn* = x*n* = ξ. Therefore,

with a uniformly random number, we know that +rn n∈U∑3∑3*xn* \W *n*∈U\W

the simulator has already completed the simulation since SIM successfully simulates REAL without knowing for all parties ∈ W. Based on the hybrid 1 to 9, we can infer that the distribution of this hybrid is identical to the real output. Completing the proof.*xn n*

VII. PERFORMANCE EVALUATION

We recruit 600 mobile devices to evaluate the performance of our VerifyNet, where most smart devices come with 4GB of RAM and are equipped with Android 6.0 systems. Each mobile device runs the same convolutional neural network offline to obtain the local gradients of all parameters. All the raw data are selected from MNIST database (http://yann.lecun.com/exdb/mnist/) which has a training set of 60,000 examples, and a test set of 10,000 examples. Besides, the “ Cloud ” is simulated with a Lenovo server which has Intel(R) Xeon(R) E5-2620 2.10GHZ CPU, 16GB RAM, 256SSD, 1TB mechanical hard disk and runs on the Ubuntu 18.04 operating system. More specifically, we adopt the key agreement protocol based on Elliptic-Curve to achieve the key distribution between two users, and the standard Shamir&apos;s out-of-N secret sharing protocol [?] to generate the shares of secret. In addition, we use AES in counter mode and AES-GCM with 128-bit keys to achieve the authenticated encryption and pseudorandom generator, respectively.*t*

## A. Functionality

TABLE I: Comparison of Functionality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PPML | PPDL | SafetyNets | VerifyNet |
| Data Privacy | X | X | × | X |
| Robustness to Failures | X | × | × | X |
| Verifiability |  |  | X | X |

                                               × ×

As shown in TABLE.I, we compare the functionality with the latest work PPML [?], PPDL [?] and SafetyNets [?], since the main works of these schemes are similar to ours. Specifically, we know that both PPML and PPDL guarantee the confidentiality of data privacy during the execution, however, the property of verifiability is not supported by their model. In addition, PPDL is also failure to deal with the problem of users dropping out. On the other hand, SafetyNets is primarily designed from the verifiability perspective, hence the problems of data privacy leakage and users dropping out in the training process are not considered in its protocol. Compared with these schemes, our VerifyNet supports each user to verify the results returned by the cloud server while guaranteeing the confidentiality of user&apos;s local gradients. Besides, similar to PPML, VerifyNet is also robust to users dropping out at any subprocess of whole work process.

## B. Classification Accuracy

In this section, we select data from MNIST database to test the classification accuracy of our VerifyNet. The experiments were conducted on a CNN network [?], [?], which consists of two convolutional layers and two fully connected layers with

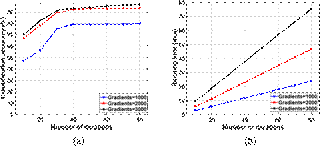


Fig. 5: (a) No dropout, |U|=100, classification accuracy with the different number of gradients per user. (b) No dropout,

|U|=100, running time with the different number of gradients per user.

128 neurons each layer. Definitely, in federated learning, the accuracy of model&apos;s outputs is closely related to two factors, i.e., the number of users participating in the training and the size of the local gradients owned by each user. In general, the accuracy of the model&apos;s output is proportional to the number of gradients/users involved in the training, and also proportional to the computation and communication overhead generated by the system. To analyze the relationship between these factors, we record the classification accuracy and running time of our VerifyNet under different number of users/gradients per user. Here we use the symbol |U| and |G| to indicate the number of users and gradients per user in our experiments, respectively.

Fig. 5 shows the classification accuracy and running time with the different number of gradients per user, where an iteration means that a parameter update (i.e., Round 0 to Round 4) is completed. For simplicity, here we only consider the case of no users dropping out. Clearly, the increase in the number of gradients facilitates the higher accuracy of the model output, but it also incurs more computation overhead (shown in Fig. 5(b)). However, by comparing classification accuracy with different gradients (See gradients=2000 and gradients=3000, respectively), the number of gradients involved in training is not the more the better, because the accuracy of the model will converge when the number of gradients increases to a certain amount. Therefore, in practical applications, we can empirically choose the appropriate number of gradients to avoid unnecessary overhead. Fig. 6 shows the classification accuracy and running time with the different number of users. Similarly, increasing the number of users in the system is beneficial to improve the model&apos;s classification accuracy, but it also requires additional computation overheads. Note that for the sake of simplicity, we do not record the total amount of data transmitted in the system under different numbers of users/gradients. However, since VerifyNet is an interactive protocol, in theory, our scheme will inevitably generate a certain communication overhead as the number of users/gradients increases.

## C. Verification Accuracy

As discussed before, to prevent malicious spoofing by

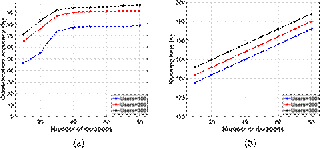


Fig. 6: (a) No dropout, |G|=1000, classification accuracy with the different number of users. (b) No dropout,

|G|=1000, running time with the different number of users.

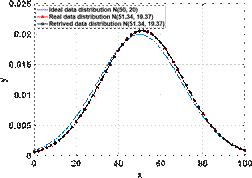


Fig. 7: Verification accuracy

the server, our VerifyNet supports each user to verify the correctness of the results returned by the server. In specific, the cloud server is required to provide the Proof about the correctness of its aggregated results to each user, and each user can reject or accept the results by checking the Proof. To give a simple presentation for the verification accuracy, we simulate 200 users uploading encrypted local data to the server, where all the data are randomly selected from normal distribution (50,20). For simplicity, here we also only consider the case of no users dropping out. Since the randomly selected data are discrete points, their real distribution ((51.34,19.37), red line in Fig. 7) is slightly different from the original ideal distribution. Then, we require the cloud server to calculate the aggregated results along with corresponding Proof for each user. If the verification is passed, the distribution of uploaded data used to generate the aggregated results should be the same as the real data. Based on this, we further use the aggregated results to calculate the mean and variance of the uploaded data. As shown in Fig. 7, we can find that retrieved data distribution is exactly overlapping with the real data distribution, which also confirms that a result returned from the server is correct once its Proof is verified.*NN*

## D. Probability of Users Dropping Out

Our VerifyNet is robust to users dropping out in training

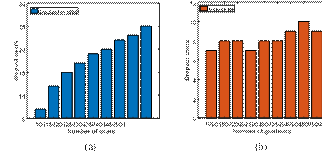
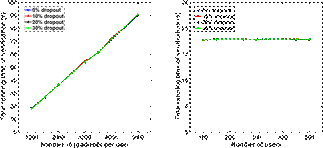


Fig. 8: Dropout users. (a) |G|=1000, with the different number of users. (b) |U|=100, with the different number of gradients per user.



                           (a) (b)

Fig. 9: Total running time of each user (Verification process).

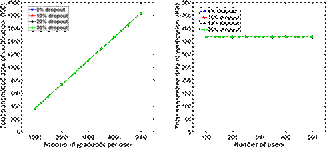
(a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users.

process, because users dropping out is very common due to users&apos; device battery issues, hardware quality problems and the like occurring in workflow. To evaluate the universality of this phenomenon, we record the number of users who logged out under different number of users/gradients per user in whole system. In specific, we require all users to upload data to the server multiple times within the specified time, and record the average of dropout users under repeated experiments. As shown in Fig. 8, we find that as the number of users/gradients increases, a certain number of users dropping out are inevitable, which is more pronounced as the number of users increases. However, Fig. 8 shows that the proportion of users dropping out is not significant relative to the total number of users. Hence, this also provides a basis for using the secret sharing protocol to manage the problem of users dropping out.

## E. Performance Analysis of Client

We analyze the performance of the client from both computation and communication overhead, where we test VerifyNet under different proportions of users dropping out.

*1) Computation Overhead:* Fig. 9 shows the running time of each user during verification process. Clearly, the user&apos;s



|  |
| --- |
| (a) (b) (c) (d)  Fig. 10: Comparison between verification computation overhead and total overhead for each user. (a) No dropout, |U|=100, with the different number of gradients per user. (b) 30% dropout, |U|=100, with the different number of gradients per user.  (c) No dropout, |G|=1000, with the different number of users. (d) 30% dropout, |G|=1000, with the different number of users. |

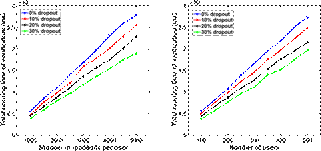
                           (a) (b)

Fig. 11: Total transmitted data of each user (Verification process). (a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users.

running time increases linearly with the increasing of the number of gradients, but keeps a constant as the number of users increases. One of main reasons is that the verification overhead is only related to the number of gradients owned by each user, since each user needs to generate (An,Bn,Ln,Qn) for newly added gradient . Fig.10 shows the comparison between the computation overhead of verification and the total overhead. For simplicity, we consider no user dropping out and 30% users dropping out in our experiments. We can see that the main computation cost of each user comes from the verification process, regardless of the number of users or gradients. In addition, our VerifyNet maintains good performance in terms of computation overhead. For example, when the number of users is 500 and the total number of gradients in our system is 500000, each user only needs about 17 seconds to complete one iteration of parameter update.*n xn*

*2) Communication Overhead:* Fig. 11 shows the total transmitted data of each user during verification process. Similar to Fig.9, the user&apos;s total transmitted data also increases linearly with the increasing of the number of gradients, but keeps a constant as the number of users increases. Fig.12 shows the

**4                                                                                                                                                                        4**



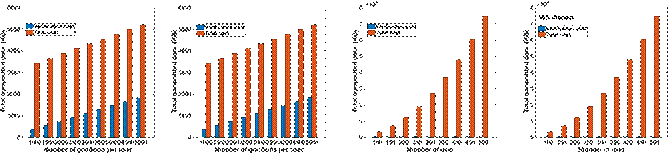
                           (a) (b)

Fig. 13: Total running time of the cloud server (Verification process). (a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users.

comparison between verification communication overhead and total overhead of each user. We can see that the proportion of overhead generated in verification process is not obvious to total overhead, and even can be ignored as the number of users increases. Moreover, experiments demonstrate that our VerifyNet still maintains good performance in terms of communication overhead. For instance, when the number of users is 500 and the total number of gradients in our system is 500000, each user only needs about 70MB to complete one iteration of parameter update.

## F. Performance Analysis of Server

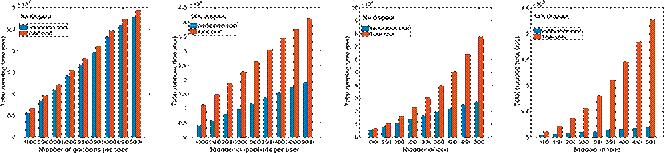
*1) Computation Overhead:* Fig.13 shows the running time of verification process of the cloud server. We can see that the server&apos;s running time increases linearly with the increasing of the number of gradients or users. The main reason is that as the number of gradients or users increases, the cloud server needs to generate the Proof of aggregated result for each new added gradients and users. Fig. 14 shows the comparison between verification computation overhead and total overhead of the cloud server. We can find that the proportion of users dropping



### (a) (b) (c) (d)

Fig. 12: Comparison between verification communication overhead and total overhead for each user. (a) No dropout, |U|=100, with the different number of gradients per user. (b) 30% dropout, |U|=100, with the different number of gradients per user.

(c) No dropout, |G|=1000, with the different number of users. (d) 30% dropout, |G|=1000, with the different number of users.



### (a) (b) (c) (d)

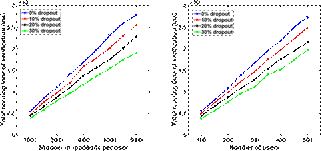
Fig. 14: Comparison between verification computation overhead and total overhead for the cloud server. (a) No dropout,

|U|=100, with the different number of gradients per user. (b) 30% dropout, |U|=100, with the different number of gradients per user. (c) No dropout, |G|=1000, with the different number of users. (d) 30% dropout, |G|=1000, with the different number of users.

out greatly determine the trend of overall cost of the cloud server, which is also obvious by comparing with Fig.14(c) and Fig.14(d). For example, when no user dropouts, the cloud sever only needs about 75000ms to complete an iteration of parameter updates, but it will take 220000ms if 30% of users dropout.

*2) Communication Overhead:* Fig. 15 shows the total transmitted data of verification process of the cloud server. We can see that as the number of users or gradients increases, the communication overhead of the cloud server also grows linearly. TABLE.I and TABLE.II show the computation and communication overhead of each round, respectively, where the red font indicates the overhead during the verification process. For each user, both the computational and communication overhead are mainly from the Masked Input and Verification, since each user needs generating and verifying Proof for each gradient, and sending the encrypted results to the cloud server. For the cloud server, after receiving all the messages on the Masked Input round, it needs to aggregate the encrypted gradients of all users and restore the secrets of all the online users in the Unmasking round, which results in large computational/*n σn*

**4                                                                                                                                                                        4**



                           (a) (b)

Fig. 15: Total transmitted data of the cloud server

(Verification process). (a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users.

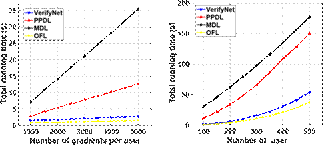
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (a) (b) (c) (d)  Fig. 16: Comparison between verification communication overhead and total overhead for the cloud server. (a) No dropout,  |U|=100, with the different number of gradients per user. (b) 30% dropout, |U|=100, with the different number of gradients per user. (c) No dropout, |G|=1000, with the different number of users. (d) 30% dropout, |G|=1000, with the different number of users.  TABLE II: Computation Overhead of Each Round   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | Dropout | Key Sharing | Masked Input | Unmasking | Verification | Total | | Client | 0% | 2508(ms) | (1311 + 11919) (ms) | 4(ms) | 5941(ms) | 21683 (ms) | | Client | 10% | 2510(ms) | (1263 + 11942)(ms) | 4(ms) | 5830 (ms) | 21549 (ms) | | Client | 20% | 2492(ms) | (1250 + 11892)(ms) | 4(ms) | 5942 (ms) | 21580 (ms) | | Client | 30% | 2480(ms) | (1245 + 11971) (ms) | 4(ms) | 5942 (ms) | 21642 (ms) | | Server | 0% | 0(ms) | 0 (ms) | (50136 + 27311)(ms) | 0 (ms) | 77477 (ms) | | Server | 10% | 0(ms) | 0 (ms) | (112379 + 24799)(ms) | 0 (ms) | 137178 (ms) | | Server | 20% | 0(ms) | 0 (ms) | (163551 + 21519)(ms) | 0 (ms) | 185070 (ms) | | Server | 30% | 0(ms) | 0 (ms) | (207222 + 19705)(ms) | 0 (ms) | 226927 (ms) |   TABLE III: Communication Overhead of Each Round   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | Dropout | Key Sharing | Masked Input | Unmasking | Verification | Total | | Client | 0% | 274(KB) | (74002 + 184) (KB) | 65(KB) | (4 + 184)(KB) | 73 (MB) | | Client | 10% | 273(KB) | (73998 + 184) (KB) | 65(KB) | (4 + 184)(KB) | 73 (KB) | | Client | 20% | 274(KB) | (73999 + 184)(KB) | 65(KB) | (4 + 184) (KB) | 73 (MB) | | Client | 30% | 274(KB) | (73998 + 184)(KB) | 65(KB) | (4 + 184) (KB) | 73 (MB) | | Server | 0% | 72(MB) | (111 + 90) (MB) | (32 + 0.18)(MB) | 0 (MB) | 305*.*18 (MB) | | Server | 10% | 72(MB) | (110 + 81) (MB) | (29 + 0.18)(MB) | 0 (MB) | 292*.*18 (MB) | | Server | 20% | 72(MB) | (102 + 72) (MB) | (25 + 0.18)(MB) | 0 (MB) | 271*.*18 (MB) | | Server | 30% | 72(MB) | (98 + 63) (MB) | (22 + 0.18)(MB) | 0 (MB) | 255*.*18 (MB) | |

communication overheads.

## G. Performance Analysis by Comparing with Existing Approaches

In this section, we analyze the cost of VerifyNet by comparing with the state-of-the-art approaches MDL [?], PPDL [?], SafetyNets [?] and Original Federated Learning model OFL[?], where OFL is the original model for performing federated learning in the plaintext environment. Here we use OFL to describe the performance differences between federated learning in plaintext and ciphertext. MDL [?] and PPDL [?] are consistent with the scenarios considered in our VerifyNet, and their goal is also to protect the privacy of user&apos;s local gradients by using privacy protection techniques. However, they do not consider the verifiability issue of the results returned by the server. Conversely, SafetyNets aims [?] to verify the correctness of results returned from the cloud, which is the first approach for verifiable execution of deep neural networks on untrusted cloud. It can convert certain types of deep neural networks into arithmetic circuits, and then verify the correctness of returned results through multiple interactions with the server.

*1) Performance of Encryption Process:* As shown in Fig.17 and Fig.18, we record the running time and total transmitted data of VerifyNet, MDL, PPDL and OFL with different number of users/gradients per user. Since verifiable calculations



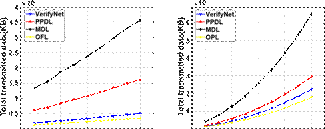
                           (a) (b)

Fig. 17: Running time. (a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users

are not considered in MDL and PPDL, the computation and communication overhead required for verification are also excluded from our VerifyNet. In addition, here we only consider the case of no users dropping out because MDL and PPDL are not supportive for users dropping out in training process. We can see that the cost of VerifyNet is significantly smaller than MDL and PPDL, while not much larger than the original solution OFL. This is mainly due to the high efficiency of our double-masking protocol compared with technologies used in MDL and PPDL. Specifically, ElGamal cryptosystem [?] is used in MDL to encrypt users&apos; local gradients, while guaranteeing multiplicative homomorphism over encrypted domain. However, since encrypting each gradient involves multiple exponential operations, ElGamal is not suitable for federated learning that is driven by large-scale data. LWEbased homomorphic encryption [?] is exploited in PPDL, which is faster than ElGamal cryptosystem. However, its computation/ communication overhead also grows significantly as the number of users/gradients per user increases. Compared with MDL and PPDL, we design a double-masking protocol to encrypt users&apos; local gradients. Since we do not consider the case of users dropping out in training process, each user *n* only needs to calculate the shares of once. As a result, the encryption operation is equivalent to adding several random values to each gradient, which greatly reduces the computation and communication overhead in the encryption process.

*2) Performance of Verification Process:* By comparing with the works SafetyNets [?] and OFL[?], we evaluate the performance of VerifyNet during the verification process. The scenario implemented in SafetyNets is different from our VerifyNet, which aims to verify the correctness of the results returned by the server during the prediction process, and only considers the number of users in whole system is 1. In order to compare the overhead in the same experimental environment, we set |U|=1, and exploit the verifiable technologies of SafetyNets to accomplish the same task of our VerifyNet. In addition, here we only consider the case of no users dropping out. Then, we record the running time and total transmitted data of three schemes with different number of gradients per user. Fig.19 shows that the cost of VerifyNet and SafetyNets

**5                                                                                    5**



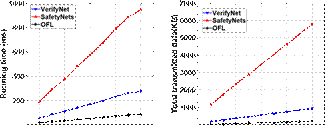
**0                                                                                    0**

**1000     2000      3000     4000      5000                          100       200        300        400       500**

### Number of gradients per user Number of users

                           (a) (b)

Fig. 18: Total transmitted data. (a) |U|=100, with the different number of gradients per user. (b) |G|=1000, with the different number of users



**0                                                                                    0**

**1000     2000     3000     4000     5000                           1000     2000     3000     4000     5000**

### Number of gradients per user Number of gradients per user

                           (a) (b)

Fig. 19: (a) |U|=1, with the different number of gradients per user. (b) |U|=1, with the different number of gradients per user.

are significant compared with original model OFL. However, the performance of our VerifyNet is significantly better than SafetyNets. One reason for this is the technical limitations of SafetyNets, and the other reason is the combination of Homomorphic Hash and pseudo-random functions exploited in our proposed protocol. Specifically, SafetyNets uses the Interactive Proof Systems [?] to check the correctness of the calculated result returned by the cloud server. It requires multiple interactions and calculations with the server to complete the verification task, and has been shown to be flawed in computation and communication overheads [?]. However, we exploit the homomorphic hash function integrated with pseudorandom technology as the underlying structure of VerifyNet, which are well known for efficiently processing of data. Hence, our VerifyNet can ensure users to verify the correctness of results returned by the cloud server with relatively low overhead.

VIII. RELATED WORKS

In this section, we introduce the latest related works of deep learning in terms of privacy protection and verifiability.

## A. Privacy-Preserving Deep Learning

Most deep learning-based privacy protection algorithms focus on protecting users&apos; data privacy. The main tools used in their protocols are differential privacy, secure multi-party computing [?], [?], and cryptographic primitives [?]. However, the issue of privacy leakage is still not completely addressed. For example, Shokri et al. [?] proposed a privacy-preserving deep learning approach by utilizing differential privacy to achieve the balance between security and accuracy. Unfortunately, any differential privacy-based strategy has been exposed to be insecure [?] if adversaries utilize the GAN network to attack the protocol. Trieu Phong et al. [?] proposed a more secure deep learning system by utilizing additively homomorphic encryption and asynchronous stochastic gradient, but the implementation requires all users to share the same key for expected security level. Recently, Keith Bonawitz et al. [?] designed a federated deep learning approach utilizing secure multi-party computing to protect the local gradient of each user, which is supportive for users offline during the training process.

## B. Verifiable Deep Learning

In deep learning, the cloud server may return incorrect results to the user due to unexpected situations. To combat that, Several schemes [?], [?] have been successively proposed to alleviate this problem. For example, Zahra Ghodsi et al. [?] designed a framework called SafetyNets. It uses the Interactive Proof Systems [?] to check the correctness of the calculated result returned by the cloud server. Later, Florian Tramr et al. [?] proposed a verifiable scheme called Slalom to perform verification by exploiting trusted hardware such as SGX, TrustZone and Sanctum. However, these schemes either support a small variety of activation functions or require additional hardware assistance. More notably, to the best of our knowledge, for a neural network being trained, there is no solution which supports the verifiability to the correctness of computation results from the cloud. Compared with existing approaches, we propose VerifyNet, the first privacypreserving approach supporting verification in the process of training neural networks. We first utilize homomorphic hash function integrated with pseudorandom technology to support the verifiability for each user. Then, we use a variant of secret sharing technology along with key agreement protocol to protect the privacy of users&apos; local gradients, and deal with the users dropping out problem during the training process.

# IX. CONCLUSION

In this paper, we have proposed VerifyNet which supports verification of the server&apos;s calculation results to each user. Besides, VerifyNet is supportive for users dropping out in training process. Security analysis shows the high security of our VerifyNet under the honest but curious security setting. In addition, experiments conducted on real data also demonstrate the practical performance of our proposed scheme. As part of future research work, we will focus on reducing the communication overhead of the entire protocol.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China under Grants 2017YFB0802300 and 2017YFB0802000, the National Natural Science Foundation of China under Grants 61802051, 61772121, 61728102, and 61472065, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2015J056.

## REFERENCES

[1] K. Ahiska, M. K. Ozgoren, and M. K. Leblebicioglu, “Autopilot design for vehicle cornering through icy roads,” IEEE Transactions on Vehicular Technology, vol. 67, no. 3, pp. 1867–1880, 2018.

[2] G. Xu, H. Li, and R. Lu, “POSTER:practical and privacy-aware truth discovery in mobile crowd sensing systems,” in Proceedings of ACM CCS, 2018, pp. 2312–2314.

[3] P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacypreserving machine learning,” in proceedings of IEEE S&P, 2017, pp. 19–38.

[4] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, “Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems,” IEEE Transactions on Industrial Informatics, 2018.

[5] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, “Efficient and privacypreserving truth discovery in mobile crowd sensing systems,” IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3854–3865, 2019.

[6] A. Datta, M. Fredrikson, G. Ko, P. Mardziel, and S. Sen, “Use privacy in data-driven systems: Theory and experiments with machine learnt programs,” in proceedings of ACM CCS, 2017, pp. 1193–1210.

[7] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, “Enabling efficient and geometric range query with access control over encrypted spatial data,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 4, pp. 870–885, 2019.

[8] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, “Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation,” IEEE Transactions on Information Forensics and Security, vol. 12, no. 3, pp. 676–688, 2017.

[9] G. Xu, H. Li, C. Tan, D. Liu, Y. Dai, and K. Yang, “Achieving efficient and privacy-preserving truth discovery in crowd sensing systems,” Computers & Security, vol. 69, pp. 114–126, 2017.

[10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” in Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in Proceedings of ACM CCS, 2017, pp. 1175–1191.

[12] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, “Trojaning attack on neural networks,” in proceedings of NDSS, 2018.

[13] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” in proceedings of NDSS, 2018.

[14] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the GAN:´ information leakage from collaborative deep learning,” in proceedings of ACM CCS, 2017, pp. 603–618.

[15] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in proceedings of IEEE S&P, 2017, pp. 3–18.

[16] Z. Ghodsi, T. Gu, and S. Garg, “Safetynets: Verifiable execution of deep neural networks on an untrusted cloud,” in Advances in Neural Information Processing Systems, 2017, pp. 4672–4681.

[17] F. Tramer and D. Boneh, “Slalom: Fast, verifiable and private execution of neural networks in trusted hardware,” arXiv preprint arXiv:1806.03287, 2018.

[18] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proceedings of ACM CCS, 2015, pp. 1310–1321.

[19] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, “Privacypreserving deep learning via additively homomorphic encryption,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1333–1345, 2018.

[20] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in Proceedings of COMPSTAT. Springer, 2010, pp. 177–186.

[21] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” in Advances in neural information processing systems, 2011, pp. 693–701.

[22] H. Li, D. Liu, Y. Dai, and T. H. Luan, “Engineering searchable encryption of mobile cloud networks: when qoe meets qop,” IEEE Wireless Communications, vol. 22, no. 4, pp. 74–80, 2015.

[23] H. Li, D. Liu, Y. Dai, T. Luan, and S. Yu, “Personalized search over encrypted data with efficient and secure updates in mobile clouds,” IEEE Transactions on Emerging Topics in Computing, vol. 6, no. 1, pp. 97– 109, 2018.

[24] A. Yun, J. H. Cheon, and Y. Kim, “On homomorphic signatures for network coding,” IEEE Transactions on Computers, vol. 59, no. 9, pp. 1295–1296, 2010.

[25] D. Fiore, R. Gennaro, and V. Pastro, “Efficiently verifiable computation on encrypted data,” in Proceedings of the ACM CCS, 2014, pp. 844–855. [26] A. Shamir, How to share a secret. Communications of the ACM, 1979.

[27] W. Diffie and M. E. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, 1976.

[28] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, “Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage,” IEEE Transactions on Emerging Topics in Computing, vol. 3, no. 1, pp. 127–138, 2015.

[29] M. Blum and S. Micali, “How to generate cryptographically strong sequences of pseudo-random bits,” in Symposium on Foundations of Computer Science, 2008, pp. 112–117.

[30] P. Rewagad and Y. Pawar, “Use of digital signature with diffie hellman key exchange and aes encryption algorithm to enhance data security in cloud computing,” in Proceedings of the IEEE CSNT, 2013, pp. 437– 439.

[31] M. Bellare and C. Namprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm,” Journal of Cryptology, vol. 21, no. 4, pp. 469–491, 2008.

[32] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in Advances in Cryptology — CRYPTO 2001, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 213–229.

[33] X. Zhang, S. Ji, H. Wang, and T. Wang, “Private, yet practical, multiparty deep learning,” in Proceedings of IEEE ICDCS. IEEE, 2017, pp. 1442– 1452.

[34] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” IEEE transactions on information theory, vol. 31, no. 4, pp. 469–472, 1985.

[35] Z. Brakerski, C. Gentry, and S. Halevi, “Packed ciphertexts in lwebased homomorphic encryption,” in International Workshop on Public Key Cryptography. Springer, 2013, pp. 1–13.

[36] J. Thaler, “Time-optimal interactive proofs for circuit evaluation,” in Advances in Cryptology - CRYPTO, 2013, pp. 71–89.

[37] J. Keuffer, R. Molva, and H. Chabanne, “Efficient proof composition for verifiable computation,” in European Symposium on Research in Computer Security. Springer, 2018, pp. 152–171.

Guowen Xu (S&apos;15) received his B.S. degree in information and computing science from Anhui University of Architecture in 2014. Currently, he is a Ph.D. student at the School of Computer Science and

Engineering, University of Electronic Science and Technology of China , China. His research interests include Cryptography, Searchable Encryption, and the Privacy-preserving Deep Learning.

Hongwei Li (M&apos;12-SM&apos;18) is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. Dr. Li serves as the Associate Editor of IEEE Internet of Things Journal, and Peer-to-Peer Networking and Applications, the Guest Editor of IEEE Network and IEEE Internet of Things Journal. He also serves/served the technical symposium cochair of ACM TUR-C 2019, IEEE ICCC 2016, IEEE

GLOBECOM 2015 and IEEE BigDataService 2015,

and many technical program committees for international conferences, such as IEEE INFOCOM, IEEE ICC and IEEE GLOBECOM. He won the Best Paper Award from IEEE MASS 2018 and IEEE HELTHCOM 2015. He is the Senior Member of IEEE, Distinguished Lecturer of IEEE Vehicular Technology Society.

Sen Liu (S&apos;17) received the BS degree in information security from Guizhou University in 2017. Currently, he is working toward the Masters degree at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include Cryptography, Searchable Encryption.

Kan Yang(M&apos;13) received the B.Eng. degree in information security from the University of Science and Technology of China in 2008 and the Ph.D. degree in computer science with Outstanding Research Thesis Award from the City University of Hong

Kong in 2013. He is an Assistant Professor with the

Department of Computer Science and the Associate

Director of the Center for Information Assurance, University of Memphis. His research interests focus on security and privacy in cloud computing, big data,

Internet of Things, information-centric network, and

distributed systems.

Xiaodong Lin (M&apos;09-SM&apos;12-F&apos;17) received the

PhD degree in Information Engineering from Beijing University of Posts and Telecommunications, China, and the PhD degree (with Outstanding Achievement in Graduate Studies Award) in Electrical and Computer Engineering from the University of Waterloo, Canada. He is currently an associate professor in the School of Computer Science at the University of Guelph, Canada. His research interests include computer and network security, privacy protection, applied cryptography, computer forensics, and soft-

ware security. He is a Fellow of the IEEE.