计算机科学
COMPUTER SCIENCE

轻量级异构安全函数计算加速框架

赵川, 何章钊, 王豪, 孔繁星, 赵圣楠, 荆山

引用本文

赵川, 何章钊, 王豪, 孔繁星, 赵圣楠, 荆山. 轻量级异构安全函数计算加速框架[J]. 计算机科学, 2025, 52(4): 301-309.

ZHAO Chuan, HE Zhangzhao, WANG Hao, KONG Fanxing, ZHAO Shengnan, JING Shan. Lightweight Heterogeneous Secure Function Computing Acceleration Framework [J]. Computer Science, 2025, 52(4): 301-309.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240600046

轻量级异构安全函数计算加速框架

赵 川[1] 何章钊[1,2] 王 豪[1,2] 孔繁星[1,2] 赵圣楠[1] 荆 山[2,3]
1 泉城省实验室 济南 250103
2 济南大学信息科学与工程学院 济南 250022
3 山东省网络环境智能计算技术重点实验室(济南大学) 济南 250022
(ise_zhaoc@ujn.edu.cn)

摘 要 当前,数据已成为关键战略资源,数据挖掘和分析技术在各行业发挥着重要作用,但也存在着数据泄露的风险。安全函数计算(Secure Function Evaluation,SFE)可以在保证数据安全的前提下完成任意函数的计算。Yao协议是一种用于实现安全函数计算的协议,该协议在混淆电路(Garbled Circuit,GC)生成和计算阶段含有大量加解密计算操作,且在不经意传输(Oblivious Transfer,OT)阶段具有较高的计算开销,难以满足复杂的现实应用需求。针对Yao协议的效率问题,基于现场可编程门阵列(Field Programmable Gate Array,FPGA)的异构计算对Yao协议进行加速,并结合提出的轻量级代理不经意传输协议,最终设计出轻量级异构安全计算加速框架。该方案中,混淆电路生成方和代理计算方都实现了CPU-FPGA异构计算架构。该架构借助CPU擅长处理控制流的优势和FPGA的并行处理优势对混淆电路生成阶段和计算阶段进行加速,提高了生成混淆电路和计算混淆电路的效率,减轻了计算压力。另外,相比于通过非对称密码算法实现的不经意传输协议,在轻量级代理不经意传输协议中,混淆电路生成方和代理计算方只需执行对称操作,代理计算方即可获取用户输入对应的生成方持有的随机数。该轻量级代理不经意传输协议减轻了用户和服务器在不经意传输阶段的计算压力。实验证明,在局域网环境下,与Yao协议的软件实现(TinyGarble框架)相比,该方案的计算效率至少提高了128倍。

关键词:安全函数计算;现场可编程门阵列;混淆电路;不经意传输;异构计算
中图分类号 TP309

Lightweight Heterogeneous Secure Function Computing Acceleration Framework

ZHAO Chuan[1],HE Zhangzhao[1,2],WANG Hao[1,2],KONG Fanxing[1,2],ZHAO Shengnan[1] and JING Shan[2,3]
1 Quan Cheng Laboratory,Jinan 250103,China
2 School of Information Science and Engineering,University of Jinan,Jinan 250022,China
3 Shandong Provincial Key Laboratory of Network Based Intelligent Computing,University of Jinan,Jinan 250022,China

Abstract Currently,data has become a crucial strategic resource,and data mining and analysis technologies play an important role in various industries. However,there are risks of data leakage in the process of data mining and analysis. Secure function evaluation(SFE in short) can perform computation of arbitrary functions while ensuring data security. Yao's protocol is a protocol used for secure function computation,which involves a significant amount of encryption and decryption operations in the garbled circuit(GC) generation and evaluation phases. It has high computational overhead in the oblivious transfer(OT) phase,making it challenging to meet the demands of complex real-world applications. Aimed at the efficiency issues of Yao's protocol,heterogeneous computing based on field programmable gate array(FPGA) accelerates the Yao's protocol and combines the proposed lightweight proxy oblivious transfer protocol,ultimately designing a lightweight heterogeneous secure computation acceleration framework. In this solution,a CPU-FPGA heterogeneous computing architecture is implemented for both the garbled circuit generation and the proxy computation tasks. This architecture leverages the advantages of CPU in handling control flow and the parallel pro-

到稿日期:2024-06-05 返修日期:2024-08-31
基金项目:国家自然科学基金(62472252,62172258);泰山学者工程(tsqn202211280);山东省自然科学基金(ZR2024QF131,ZR2023LZH014,ZR2022ZD01,ZR2022MF264,ZR2021LZH007);山东省重点研发计划(2021SFGC0401,2021CXGC010103);山东省科学技术厅(SYS202201);泉城省实验室重大项目(QCLZD202302)
This work was supported by the National Natural Science Foundation of China (62472252, 62172258), Taishan Scholars Program (tsqn202211280), Shandong Provincial Natural Science Foundation (ZR2024QF131, ZR2023LZH014, ZR2022ZD01, ZR2022MF264, ZR2021LZH007), Key R&D Program of Shandong Province(2021SFGC0401,2021CXGC010103), Department of Science & Technology of Shandong Province(SYS202201) and Quan Cheng Laboratory(QCLZD202302).
通信作者:赵圣楠(zsn.sdu@gmail.com)

cessing capabilities of FPGA to accelerate the garbled circuit generation and evaluation phases, increasing the efficiency of generating and evaluating garbled circuits and reducing computational pressure. In addition, compared to the oblivious transfer protocol implemented through asymmetric cryptographic algorithms, in the lightweight proxy oblivious transfer protocol, only symmetric operations are required for the garbled circuit generator and the proxy calculator. The proxy calculator can then obtain the random number held by the generator corresponding to the user's input. This lightweight proxy oblivious transfer protocol alleviates the computational pressure on the user and the server during the oblivious transfer phase. Experimental results show that in a local area network environment, compared to software implementation of Yao's protocol (TinyGarble framework), our solution improves computational efficiency by at least 128 times.

Keywords Secure function evaluation, Field programmable gate array, Garbled circuits, Oblivious transfer, Heterogeneous computing

1 引言

大数据时代，互联网公司通过收集用户的各类信息来提高自身服务质量，例如，谷歌、阿里、亚马逊等公司收集用户买过的商品和看过的广告等数据，通过挖掘这些数据的有效信息，有针对性地将产品推荐给用户。除此之外，数据还可以用于民生服务和社会治安，如预测房价涨势[1]，预防疾病[2]，以及侦测潜在的恐怖主义威胁[3]。但是，大数据可能会侵犯用户隐私，甚至导致隐私信息泄露。例如，宜家（IKEA）加拿大公司通报公司发生了大约 9.5 万名客户信息的泄露事件，泄露的数据包括客户的电子邮件地址、电话号码等隐私信息；美国 RR Donnelly 公司有 2.5 GB 用户数据被泄露；超星学习通的数据库有一亿多条数据信息被泄露，被公开售卖的数据中包括手机号、账号密码等个人隐私信息。这些隐私泄露事件都造成了严重的后果，给公司和用户带来了不可估量的损失，说明保护数据隐私至关重要。

最近一系列研究工作[4-9]通过密码学方案和安全技术来保护数据挖掘过程中用户的隐私。安全函数计算（Secure Function Evaluation，SFE）[10]允许不同的参与方在不揭示各自输入的前提下安全地计算出目标函数。因此，安全函数计算成为数据共享和数据挖掘的主流隐私保护计算方法之一。混淆电路（Garbled Circuit，GC）[11]是一种特殊的布尔电路构造，可以在保护计算任务参与者输入隐私性的前提下进行安全函数计算。不经意传输（Oblivious Transfer，OT）[12]是一个密码学协议，在这个协议中消息发送者从一些待发送的消息中选择一条发送给接收者，但事后却不知道发送了哪一条消息。

Yao 协议是基于混淆电路和不经意传输协议设计的，包括 3 个阶段，分别是混淆电路生成阶段、不经意传输阶段和混淆电路计算阶段。混淆电路生成和计算阶段涉及大量的加解密计算，且不经意传输通过耗时的非对称密码算法（如 RSA）实现，这些对于安全计算的参与者来说计算压力非常大。Yao 协议的整体效率问题，导致其难以有效地应用于现实场景。

为解决 Yao 协议计算效率低的问题，可采用现场可编程门阵列（Field Programmable Gate Array，FPGA）、图形处理器（Graphics Processing Unit，GPU）、专用集成电路（Application Specific Integrated Circuit，ASIC）等硬件加速器，基于硬件资源对加解密算法进行加速，从而使得计算效率远超于基于中央处理器（CPU）的软件计算效率。但不同的硬件加速器在架构和适用的计算类型上会有很大区别。例如，GPU 计算位宽固定，适合计算浮点类型的数据，但在面对不同位宽的数据计算时，GPU 无法灵活地调整计算位宽，而是以固定的几种位宽进行计算。与 GPU 相比，FPGA 可以更改电路，以适用于不同位宽数据的计算。ASIC 是专用的集成芯片，对于特定算法，该芯片的计算效率远高于 GPU 和 FPGA。然而 ASIC 内部的电路架构无法更改，无法计算其他的算法，因此 ASIC 的灵活性最差，而 FPGA 和 GPU 可以加速不同算法的计算。

异构计算[13-15]主要是指使用不同类型指令集和体系架构的计算单元组成系统的计算方式。目前主流的异构计算方式有 CPU-GPU，GPU-FPGA，CPU-FPGA 等，不同形式的异构计算具有不同的特点。例如，CPU-FPGA 异构计算同时具有 CPU 擅长串行计算和控制流处理的优势以及 FPGA 高效并行计算的优势，适用于多种类型的计算任务的加速。而 GPU-FPGA 的异构计算形式具有极高的并行计算能力，适用于处理大量且重复的工作。

本文针对安全函数计算方案的效率问题，提出了轻量级异构安全函数计算加速框架。该框架在 Yao 协议的基础上添加了代理计算方进行混淆电路计算，从而减轻了安全函数计算中参与方的计算压力。进一步地，对于生成和计算混淆电路，本文实现了一种基于 CPU-FPGA 异构计算的架构，混淆电路生成方利用该架构加速混淆电路生成，代理计算方利用该架构加速混淆电路计算，从而提高了协议整体的计算效率。

此外，为了最大化发挥该框架下 3 个参与方的优势，对不经意传输协议进行优化和改进，提出了轻量级代理不经意传输协议，缓解了安全函数计算参与者在不经意传输阶段的计算压力。

本文的主要贡献包括以下两个方面：

1）实现了一种基于 CPU-FPGA 异构计算的架构，通过该架构加速生成和计算混淆电路，提高了计算效率，减轻了安全计算参与者的计算压力；

2）提出了轻量级代理不经意传输协议，减轻了安全计算参与者在不经意传输阶段的计算压力。

2 相关工作

目前已有很多研究工作将 FPGA 用于加速计算。为了解决机器学习或安全协议计算效率低的问题，文献[16]提出

了一种基于 GPU-FPGA 的机器学习加速平台。由于 GPU 是单指令流多数据流的架构，在面对大量数据的并行计算时吞吐量比 FPGA 高，因此该平台的 GPU 负责机器学习的训练，FPGA 负责机器学习的推理。该平台还包括一个模型移植部分，可以将模型从训练部分移植到推理部分。文献[17]提出了一种由 CPU 和 FPGA 组成的异构计算系统，在异构计算系统上实现混淆电路，在不揭示数据的前提下保护函数的安全计算。在混淆表生成过程中，CPU 将混淆表的信息发送给 FPGA，FPGA 加速生成混淆电路，从而提高整体协议的执行速度。文献[18]在客户端/服务器结构中考虑安全函数计算，其中服务器向客户端颁发安全令牌。令牌不受客户端信任，也不是受信任的第三方。该文献利用令牌大幅降低安全函数计算的通信复杂性和服务器的计算负载，并使用 FPGA 进行了原型设计。

MAXelerator[19]是第一个用于在云服务器上实现隐私保护机器学习推理的硬件加速器，其为云服务器上的基于矩阵乘法的机器学习创建了一个实用的隐私保护方案。在 MAXelerator 中，云上的 FPGA 和 CPU 共同作为混淆电路生成方生成混淆电路，由 FPGA 对混淆操作加速，CPU 负责总体控制。与文献[19]相比，ReDCrypt[20]既不需要改变 AI 模型的训练方式，也不需要依赖两个非共谋的服务器来执行，能够处理更大规模的问题。文献[21]提出一种基于 CPU 和 FPGA 异构计算的可动态重新配置的架构，通过该架构加速混淆电路生成。该架构包括门混淆单元、块随机存储器(Block RAM，BRAM)、用于缓存数据的先入先出队列，以及工作负载分配器等组件，在面对不同问题时，无需重编程 FPGA，提高了可扩展性。与软件实现相比，该架构在混淆电路生成阶段实现了极高的计算效率。相比于文献[21]，文献[22]将 FPGA 开发板上的双倍速率同步动态随机存储器(Double Data Rate，DDR)加入该架构中，能够处理更大规模的问题，并且作者将通信部分和计算部分进行重叠，提高了生成混淆电路的效率。文献[23]对电路中门的混淆处理提出了一种调度方法，该方法先计算出电路中每个门的优先级，再根据门的优先级对门进行排序，并将门放入队列中，优先级高的门会进行优先加密，从而能够减少 FPGA 的空闲时间，提高效率。文献[24]提出了一个在云上部署的多方计算即服务(Multi-Party Computation as a Service，MPCaaS)，用于支持更多组织和个人安全地进行共享数据处理，并使用 FPGA 加速基于混淆电路的多方计算协议，提供更高效的服务。文献[25]提出了用于同态卷积层的现场可编程门阵列硬件加速器，针对最先进的同态卷积层算法的不同参数集和应用场景提出了 3 种高度并行的架构，并在 Xilinx VCU110 FPGA 板上实现。

3　预备知识

3.1　不经意传输协议

不经意传输是密码学中的基本原语之一，对安全计算领域的研究具有重要的作用。本文关注标准的 2 选 1 不经意传输协议。发送方 Alice 拥有 2 个有序随机串(m_0,m_1)；接收方 Bob 根据选择比特 b，选择接收其中一个随机串m_b。在协议结束后，接收方 Bob 根据自己的输入获得随机串m_b，而得不到关于m_{1-b}的任何信息。发送方 Alice 则不会获得任何输出，并且无法得知接收方 Bob 最后获得哪一个随机串。

3.2　Yao 协议

Yao 协议由混淆电路和不经意传输组成，其可以分为 3 个阶段，即混淆电路生成阶段、不经意传输阶段和混淆电路的计算阶段。经典的两方 Yao 协议交互示意图如图 1 所示。

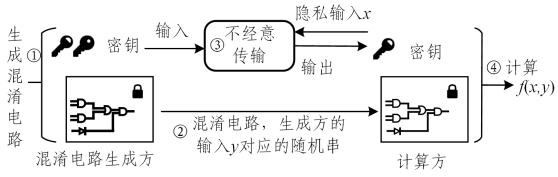


图 1　Yao 协议交互图

Fig. 1　Interaction diagram of Yao's protocol

1)混淆电路生成阶段。在步骤①中，混淆电路生成方首先使用伪随机函数为电路中的每条输入线上的每个可能的值生成一个随机串，该随机串的长度与安全参数有关。本文中，$k_{w_i}^b$ 表示线w_i上的值$b\in\{0,1\}$对应的随机串。对于每个门，混淆电路生成方利用输入线w_i和w_j的随机串，按照式(1)加密输出线对应的随机串，并根据门的逻辑情况生成相应的密文：

$$Enc_{(k_{w_i}^{b_i},k_{w_j}^{b_j},g)}(k_{w_k}^{g(b_i,b_j)})=Enc_{(k_{w_i}^{b_i})}(Enc_{(k_{w_j}^{b_j})}(k_{w_k}^{g(b_i,b_j)})) \tag{1}$$

假设w_i和w_j是与门的输入线，w_k是与门的输出线，g 是门的标识，加密与门输出线随机串的 4 种逻辑情况如表 1 所列。完成门输出线上的随机串加密之后，混淆电路生成方需要把密文的顺序进行随机置换，从而得到混淆表，随后在步骤②中将混淆电路以及自己输入对应的随机串发送给混淆电路计算方。

表 1　加密与门输出线随机串的 4 种逻辑情况

Table 1　Four logical cases of encrypting random strings of output lines of AND gates

b_i	b_j	b_i AND b_j	密文
0	0	0	$Enc_{(k_{w_i}^0,k_{w_j}^0,g)}(k_{w_k}^0)$
0	1	0	$Enc_{(k_{w_i}^0,k_{w_j}^1,g)}(k_{w_k}^0)$
1	0	0	$Enc_{(k_{w_i}^1,k_{w_j}^0,g)}(k_{w_k}^0)$
1	1	1	$Enc_{(k_{w_i}^1,k_{w_j}^1,g)}(k_{w_k}^1)$

2)不经意传输阶段。步骤③的不经意传输表示混淆电路计算方与混淆电路生成方执行不经意传输协议来获取计算方的输入所对应的随机串。例如，电路中线w_i的随机串为$k_{w_i}^0$和$k_{w_i}^1$，计算方的输入为 0，则需要获得$k_{w_i}^0$。在此过程中，计算方只能获取输入所对应的随机串，而混淆电路生成方无法获得任何内容。

3)混淆电路计算阶段。步骤④表示混淆电路计算方计算混淆电路的过程，计算方获取输入所对应的随机串后，根据式(2)解密电路第一层中门输出线的密文：

$$Dec_{(k_{w_i}^{b_i},k_{w_j}^{b_j},g)}(c)=Dec_{(k_{w_i}^{b_i})}(Dec_{(k_{w_j}^{b_j})}(c)) \tag{2}$$

其中，c 代表密文。这些密文中只有一个会被成功解密，从而得到门输出线的随机串。计算方将这个随机串作为密钥继续解密下一层电路中相应门输出线的密文。经过层层解密，计算方最终得到输出结果。

与 Yao 协议不同，代理 Yao 协议有 3 个参与方，分别是

用户、混淆电路生成方和代理计算方，如图 2 所示。代理 Yao 协议在原有 Yao 协议的基础上添加了代理计算方替代用户对混淆电路进行计算，减轻了用户的计算量。这里的用户是安全函数计算中的参与者。在代理 Yao 协议的不经意传输阶段，代理计算方与用户、混淆电路生成方共同执行代理不经意传输协议，代理计算方能够获得用户输入所对应的生成方持有的随机串。

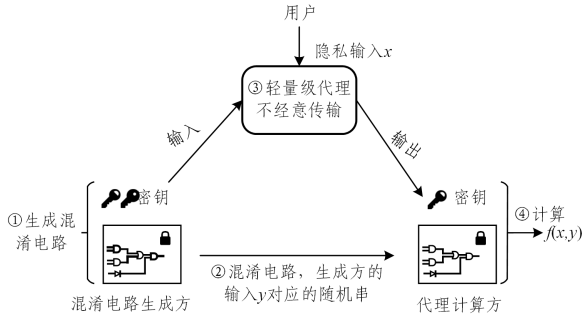


图 2 代理 Yao 协议的交互图

Fig. 2 Interaction diagram of proxy Yao's protocol

3.3 基于 FPGA 的异构计算

异构计算主要是指利用不同类型指令集和体系架构的计算单元的特点和优势，实现更高的计算效率。目前主流的异构计算方式有 CPU-GPU，CPU-FPGA，GPU-FPGA 等组合方式。异构计算旨在利用不同计算单元的特点和优势，以更高的计算效率完成不同的计算任务。接下来将举例说明基于 FPGA 的异构计算的方式和工作负载分配策略。

基于 GPU-FPGA 的异构计算根据 GPU 和 FPGA 的特点，合理分配 GPU 和 FPGA 的工作负载，可以有效地提高异构计算的效率。例如，GPU 负责机器学习训练阶段的加速，FPGA 负责机器学习推理阶段的加速，以最合理的工作负载提高异构计算系统的吞吐量。在基于 CPU-FPGA 的异构计算中，CPU 负责处理异构计算系统的通信和调度部分。例如，在机器学习推理中，CPU 负责与用户通信和传输 FPGA 计算需要的数据。另外，在基于 CPU-GPU-FPGA 的异构计算场景中，CPU 可分别辅助 GPU 和 FPGA 进行机器学习训练和机器学习推理。具体来说，CPU 负责训练和推理中的串行计算任务，GPU 和 FPGA 分别负责训练和推理中的并行计算。

4 异构安全计算加速框架

本文的框架包含 3 个参与方，分别是混淆电路生成方、代理计算方和用户。混淆电路生成方生成混淆电路，代理计算方计算混淆电路，用户提供隐私输入。混淆电路生成方和代理计算方仅协助用户进行安全计算，而不提供任何输入。该框架包含两个部分，即基于 CPU-FPGA 异构计算的架构和轻量级代理不经意传输协议。接下来将分别介绍基于 CPU-FPGA 异构计算的架构和轻量级代理不经意传输协议，以及异构安全计算加速框架的安全性分析。

4.1 基于 FPGA 的异构计算

混淆电路生成和计算的流程如图 3 所示，异构安全计算加速框架使用两台不合谋的主机分别实例化混淆电路生成方和代理计算方，生成方和计算方都部署了基于 CPU-FPGA 异构计算的架构。混淆电路的生成阶段和计算阶段分别包含大量的加密操作和解密操作，因此，利用 FPGA 加速的部分更偏向于解决计算密集带来的高延迟问题。

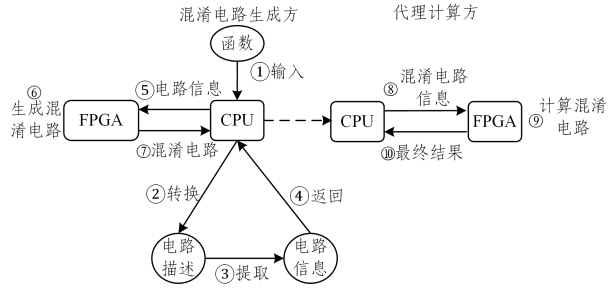


图 3 生成和计算混淆电路的流程

Fig. 3 Flow for generating and computing garbled circuit

如图 3 所示，在混淆电路生成阶段的步骤①和步骤②中，混淆电路生成方的主机利用 TinyGarble 框架[26]将函数转换为电路描述。随后，生成方主机在步骤③从电路描述中提取电路信息，并在步骤④将电路信息返回给 CPU。在步骤⑤中，CPU 将电路信息发送至 FPGA 中。在步骤⑥中，生成方的 FPGA 通过电路信息生成混淆电路，并在步骤⑦将混淆电路发送回生成方的 CPU。在混淆电路计算阶段，代理计算方的 CPU 在步骤⑧中将混淆电路发送给代理计算方的 FPGA。FPGA 在步骤⑨对混淆电路进行计算，并在步骤⑩将最终结果返回 CPU。

接下来，将按照协议的执行流程详细介绍基于 CPU-FPGA 异构计算的架构。如图 4 所示，该架构包含了 FPGA 的 DDR、工作负载分配器、BRAM、门混淆单元和 CPU 等组件。图 4 中的 AND 或 XOR 都是门混淆单元，AND 和 XOR 门混淆单元比单个位门复杂得多，是由查找表资源构成的。

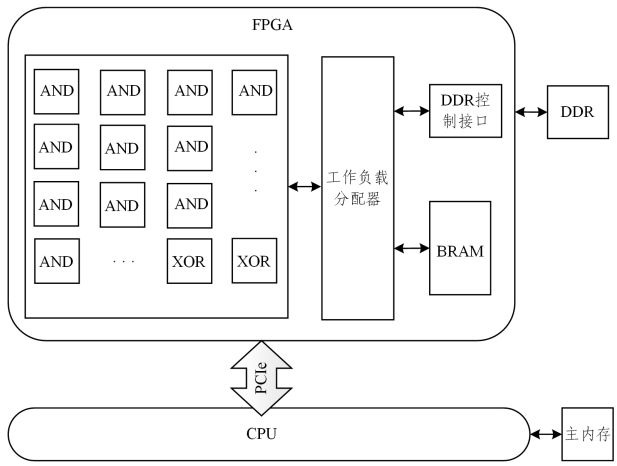


图 4 基于 CPU-FPGA 异构计算的架构

Fig. 4 CPU-FPGA heterogeneous computing based overlay architecture

使用基于 CPU-FPGA 异构计算的架构加速混淆电路生成时，首先由混淆电路生成方的 CPU 为函数生成电路描述。随后，CPU 从电路描述中提取出电路每层中的门信息，这些信息包括每层电路门的数量、每个门的标识符等。然而，电路中的门之间存在输入依赖的情况，不可能同时进行混淆处理

（混淆处理指加密门的输出线随机串并将得到的密文打乱）。因此，CPU 按照广度优先的顺序，通过 PCIe 将电路中的门信息发送到 DDR 中。接下来，FPGA 上的工作负载分配器读取 DDR 中的门电路信息，并将其存储在 BRAM 中。在进行门的混淆处理时，工作负载分配器负责协调门混淆单元处理的顺序。在与门的混淆处理中，每个与门混淆单元使用与门输入线的随机串，按照式(3)对与门的输出线随机值进行加密。

$$Enc_{(k_{w_i}^{b_i},k_{w_j}^{b_j},g)}(k_{w_k}^{g(b_i,b_j)})=SHA(k_{w_i}^{b_i}\parallel k_{w_j}^{b_j}\parallel g)\oplus k_{w_k}^{g(b_i,b_j)} \tag{3}$$

其中，SHA 是 SHA-1 哈希算法，$\parallel$ 代表级联，g 是门标识符，$\oplus$ 是异或运算。本文采取了哈希算法结合异或运算对与门输出线进行加密，因此无需同式(1)一样进行两次加密。由于与门的每条输入线都包含两个值，因此加密与门时包含 4 种逻辑情况。如果依次对这 4 种逻辑情况下的与门进行加密，需要耗费较长的时间，因此该架构根据与门的 4 种逻辑情况，在每个与门混淆单元中实现了 4 个 SHA-1 核，这 4 个 SHA-1 核以并行计算的方式进行与门的加密操作，如图 5 所示。SHA-1 核的模块图如图 6 所示。

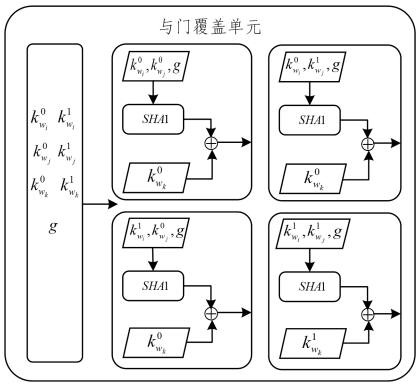


图 5　与门混淆单元

Fig. 5　Overlay AND gate

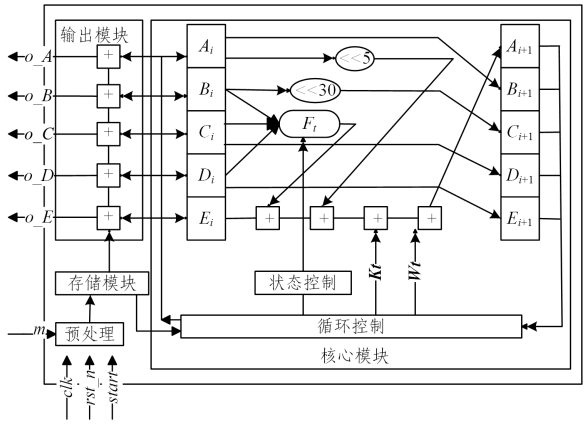


图 6　SHA-1 核模块图

Fig. 6　Module of SHA-1 core

对于异或门来说，由于该架构采用了 free-XOR[27] 技术，因此异或门的混淆处理无需加解密计算。异或门同一输入线上的两个随机串存在关系 $k_{w_i}^1=k_{w_i}^0\oplus\Delta$，$k_{w_j}^1=k_{w_j}^0\oplus\Delta$，因此 $k_{w_k}^{g(b_i,b_j)}$ 的两种情况是 $k_{w_i}^0\oplus k_{w_j}^0$ 和 $k_{w_i}^0\oplus k_{w_j}^0\oplus\Delta$，其中 Δ 是全局偏移量。因此，根据式(4)对异或门进行处理：

$$k_{w_k}^{g(b_i,b_j)}=k_{w_i}^{b_i}\oplus k_{w_j}^{b_j} \tag{4}$$

当与门和异或门的输出线随机串处理完成后，工作负载分配器将门的输出线随机串的处理结果存入 BRAM 中。在该结果作为密钥完成后续的混淆操作后，负载分配器将该结果从 BRAM 中删除。如果 BRAM 已满，则 DDR 存储剩余的结果，并且当每个门的混淆操作完成后，DDR 存储生成的混淆表。最终，FPGA 通过 PCIe 总线将混淆电路发送回 CPU，生成方 CPU 将混淆电路以及混淆电路输出线随机串和真实值的对应关系发送给代理计算方的 CPU。

代理计算方 CPU 接收到生成方发来的混淆电路并通过轻量级代理不经意传输协议获取随机串后进行混淆电路的计算，流程与混淆电路的生成类似：代理计算方的 CPU 将混淆电路和随机串发送到 FPGA 的 DDR 上，FPGA 的工作负载分配器负责读取信息，并将门的计算任务分配给门混淆单元。与门混淆单元按照式(5)对密文进行解密，得到与门输出线的随机串。

$$Dec_{(k_{w_i}^{b_i},k_{w_j}^{b_j},g)}(c)=SHA(k_{w_i}^{b_i}\parallel k_{w_j}^{b_j}\parallel g)\oplus c \tag{5}$$

该架构采用 point-and-permute[28] 技术，代理计算方计算混淆电路时每个门的输出线的密文仅需要解密一次，因此，代理计算方的每个与门混淆单元中仅有一个 SHA-1 核。异或门混淆单元根据式(4)获取异或门输出线的随机串。

4.2　轻量级代理不经意传输协议

为了解决用户在不经意传输中运算效率低的问题，在异构安全计算加速框架中提出了轻量级代理不经意传输协议。该协议有 3 个参与方，分别是混淆电路生成方、用户和代理计算方。生成方和代理计算方的对称操作可使代理计算方获取到用户输入对应的生成方持有的随机串，减少了不经意传输阶段的计算量。轻量级代理不经意传输协议如协议 1 所示。

协议 1　轻量级代理不经意传输协议

输入：混淆电路生成方持有随机串 m_0，m_1，用户持有输入 $x\in\{0,1\}$

输出：代理计算方获得随机串 m_x

预处理阶段：

1. 用户生成随机串 r_0，r_1 和随机数 b，$b\in\{0,1\}$；

2. 计算 $a=x\oplus b$；

在线阶段：

3. 用户将 a，r_0，r_1 安全发送给混淆电路生成方；

4. 用户将 b，r_b 安全发送给代理计算方；

5. 混淆电路生成方接收到 a，r_0，r_1 后计算 y_0，y_1：

$y_0=m_a\oplus r_0$，$y_1=m_{1-a}\oplus r_1$

6. 混淆电路生成方将 y_0，y_1 安全发送给代理计算方；

7. 代理计算方根据用户发送的 a，r_0，r_1 和生成方发送的 y_0，y_1 计算出 m_x：$m_x=y_b\oplus r_b$

1）正确性分析

因为随机数 $b\in\{0,1\}$，因此 b 有两个可能的值：0 或 1。当 $b=0$ 时，代理计算方计算 $y_b\oplus r_b$ 得到 m_a。而 $m_a=m_{x\oplus b}=m_x$，因此 $y_b\oplus r_b=m_x$。当 $b=1$ 时，代理计算方计算 $y_b\oplus r_b$ 得到 m_{1-a}。而 $m_{1-a}=m_{1\oplus x\oplus b}=m_x$，因此 $y_b\oplus r_b=m_x$。综上所述，通过该协议，代理计算方正确获得了用户输入 x 对应的随机串 m_x。

2）协议安全性分析

该协议在线阶段中的步骤 3 和步骤 4 安全地将用户生成

的随机串分别发送给生成方和代理计算方是为了保护用户的数据隐私，因为混淆电路生成方和代理计算方都是半诚实的，将 b 和 a 发送至不同的参与方，可防止参与方通过 $a \oplus b$ 获得用户的数据 x。而步骤 5 将随机串与 r_0 或 r_1 计算异或是为了保护生成方持有的随机串，通过该方式，代理计算方只能获得其中一个随机串，因为在步骤 4 中用户只将 b, r_b 发送到代理计算方，并没有发送 r_{1-b}。假如代理计算方获得了两个随机串，那么代理计算方可以解密生成方发送的混淆表中的每种情况，根据输出线随机串的规律有可能获取用户的数据。因此，步骤 5 可以防止代理计算方获取用户的数据。另外，该协议中的数据都是通过安全信道发送的，因此，即使数据在传输过程中被窃听，窃听者也无法得到原始数据。

不经意传输中的两方需要使用非对称密码算法实现，需要较高的计算量才能使得接收方获取到输入对应的随机串。与不经意传输协议相比，在轻量级代理不经意传输协议中，用户仅需要生成随机数，不需要进行计算，而混淆电路生成方和代理计算方仅需进行异或计算，即可使代理计算方获取到用户输入 x 对应的随机串 m_x，各个参与方的计算压力明显减小。

4.3 安全性分析

本节将对异构安全计算加速框架的安全性进行分析。该框架采取了基于 CPU-FPGA 异构计算的架构对混淆电路的生成和计算过程进行加速。在混淆电路生成阶段，生成方的架构中的 CPU 为函数生成电路描述，然后提取电路中的信息并将电路中的信息发送到 FPGA 的 DDR 上，这个过程中的数据不具有敏感信息，因此没有信息泄露问题。

FPGA 中的工作负载分配器从 DDR 中读取信息，并根据这些信息，将门的混淆处理分配给门混淆单元。负载分配器将混淆处理过程中产生的中间结果（如与门输出线随机串的密文）存入 BRAM 中，如果 BRAM 空间不够用，那么负载分配器对中间结果进行对称加密后再将其存入 DDR 中。在整个过程中，只有工作负载分配器可以读写门混淆单元的数据，即使恶意 ip 核或者 CPU 中不可信的软件堆栈企图访问 DDR 中的数据，也无法获得原始数据。门混淆单元完成混淆处理后，FPGA 将生成的混淆电路通过生成方 CPU 发送给计算方。

接下来，生成方、用户和计算方执行轻量级代理不经意传输协议。在协议执行过程中，用户无法获取到其他两方的任何信息，计算方只能获取到用户输入对应的随机串，无法获取到生成方持有的其他的随机串，而生成方也无法获取用户的输入。计算方获取到随机串以及生成方发来的混淆电路后进行混淆电路的计算，计算方内的架构采取与生成方内部同样的方式防止数据泄露。

因此，异构安全计算加速框架具有较好的隐私保护效果，可以防止 CPU 不可信软件堆栈以及恶意 ip 核泄露隐私数据。

5 实验设计与分析

本文提出基于 FPGA 的异构安全计算加速框架，用该框架加速 Yao 协议的执行，并将该框架的性能与软件实现进行了对比。本实验的器件包括主机 PC 和 Stratix V GX FPGA 开发板。主机 PC 的 CPU 是以 2.9 GHz 为基准速度的 Intel Core i7 处理器；Stratix V GX 开发板内含有 72 比特数据位宽的 DDR3 存储器，开发板上的芯片含有 622 000 个逻辑单元，512 个 18 bit×18 bit 的乘法器，50 MB BRAM 等计算资源和存储资源。在本实验中，该开发板在 200 MHz 的频率下运行。CPU 和 FPGA 之间通过 PCIE * 8 第二代总线连接，每条通道的吞吐量为 5Gbps，使该框架内部 CPU 与 FPGA 之间的通信不会成为性能瓶颈。本章实验分为生成和计算混淆电路、轻量级代理不经意传输两个实验。

5.1 生成和计算混淆电路实验

TinyGarble 框架为函数生成电路描述时，能够减少电路中非异或门的数量，提高生成和计算混淆电路的效率。例如，TinyGarble 将 1 024 bit 乘法电路中的非异或门减少了 80%，将 16 000 bit 的汉明距离的电路压缩到原来的 1/7 345，减少了 47%的非异或门。因此，本文使用 TinyGarble 框架作为混淆电路的软件实现，并在异构安全计算加速框架中使用该框架为函数生成电路描述。

如图 7 所示，架构中每个与门混淆单元进行混淆处理之前，负载分配器从 BRAM 中读取两条输入线的随机串并传入与门混淆单元中，需要两个时钟周期。与门的混处理需要 82 个时钟周期。将结果读入 BRAM 和存储到先入先出队列中都需要一个时钟周期，一共需要 86 个时钟周期。将处理结果存入先入先出队列，是为了后续将处理结果发送给 CPU。异或门的混淆处理需要 1 个时钟周期，BRAM 读写需要 3 个时钟周期，一共需要 4 个时钟周期。

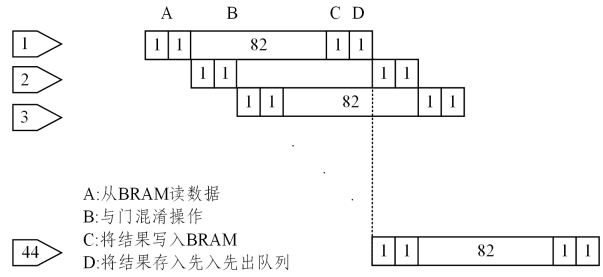


图 7 与门混淆处理序列

Fig. 7 Garbling processing sequences of AND gates

图 7 展示了与门的混淆处理序列，当第 44 个与门进入与门混淆单元之前，第一个与门混淆单元上与门的处理已经完成，因此第 44 个与门可在第一个与门混淆单元上进行处理，形成一种流水的处理序列。异或门的混淆处理序列与与门类似，但由于异或门的混淆处理和读写数据仅需要 4 个时钟周期，因此两个异或门混淆单元就可以以流水线并行的形式处理异或门。本实验在异构计算架构中实例化 43 个与门混淆单元和 2 个异或门混淆单元。代理计算方计算混淆电路时，与门的解密处理序列与图 7 相同。

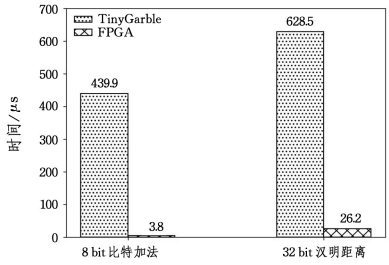
混淆电路生成方与门混淆单元占用 FPGA 资源的情况如表 2 所列，计算方的门混淆单元内部只有一个 SHA-1 核，因此，占用资源只有生成方的 1/4。本实验并没有使用 DSP 资源，因此不在表中列出 DSP 的使用率。

表 2　FPGA 的资源占用情况

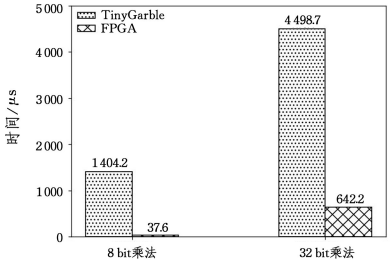
Table 2　Resource utilization of FPGA

资源	占用资源	可用资源	占用率/%
可适配逻辑块	168120	234720	72
寄存器	309960	939000	33

本文将无线局域网下异构计算架构生成混淆电路和计算混淆电路花费的时间与 TinyGarble 花费的时间进行了比较，结果如图 8 所示，本实验实现了该架构的核心功能即门混淆单元。图 8 中显示的 FPGA 的时间不包含 CPU 和 FPGA 通信的时间以及架构中负载分配器管理数据花费的时间。



(a) 8 bit 加法和 32 bit 汉明距离



(b) 8 bit 乘法和 32 bit 乘法

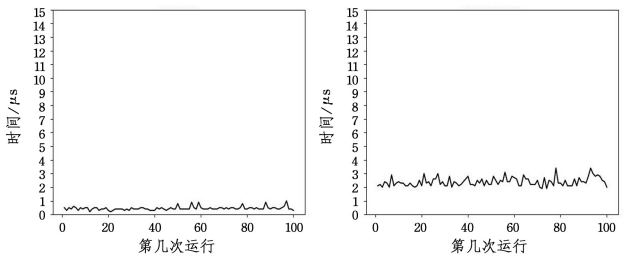
图 8　TinyGarble 与 FPGA 在生成和计算混淆电路时花费的总时间对比

Fig. 8　Comparison of TinyGarble and FPGA in the total time spent in generating and evaluating garbled circuits

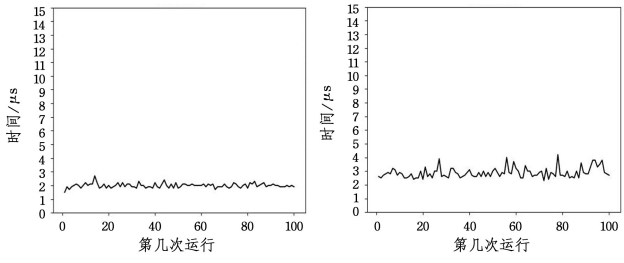
与 CPU 相比，FPGA 生成和计算混淆电路实现了较高的加速效率。然而随着应用场景复杂度的提高，FPGA 相对于 CPU 的加速比会降低。例如，图 8(a)中 8 bit 加法的加速比为 116 倍，而图 8(b)中 32 bit 乘法的加速比为 7，这是由于 FPGA 采用了 BRAM 和 DDR 混合存储机制。随着复杂度增加，BRAM 中存储的中间结果的比例越来越少，大多数据存入了 DDR 中。但是，DDR 是 FPGA 片外的存储器，读取速度较慢，因此在 32 bit 乘法的比较中，FPGA 加速比比较小。

5.2　轻量级代理不经意传输实验

图 9 给出了局域网下轻量级代理不经意传输协议的运行时间。图 9(a)为代理计算方的时间开销，图 9(b)为混淆电路生成方的时间开销，图 9(c)为协议离线阶段的时间开销，图 9(d)为协议在线阶段的时间开销。其中，离线阶段的时间开销指用户生成随机数所花费的时间，在线阶段的时间开销是代理计算方的时间开销和混淆电路生成方的时间开销的总和。横轴是第几次运行；纵轴是运行时间，单位是微秒(μs)。从图 9 可看出，代理计算方和混淆电路生成方的运行时间基本都在 0～3 μs，离线阶段时间不超过 1 μs，在线阶段的时间开销在 2～4 μs 波动。



(a) 代理计算方时间开销　(b) 混淆电路生成方时间开销

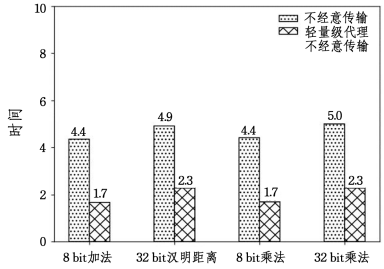


(c) 协议离线阶段时间开销　(d) 协议在线阶段时间开销

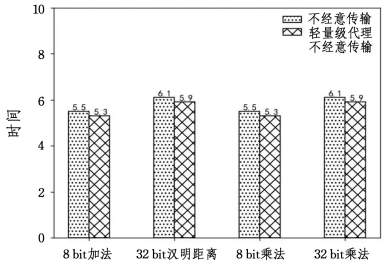
图 9　轻量级代理不经意传输协议的运行时间

Fig. 9　Runtime of the lightweight proxy oblivious transfer protocol

图 10 给出了不经意传输和轻量级代理不经意传输时间开销的对比。本实验的局域网环境中轻量级代理不经意传输协议的时间开销太小，无法用柱状图展示，因此本实验将单位为微秒(us)的时间数据取以 10 为底的对数后再将其显示在图中。



(a) 局域网



(b) 无线局域网

图 10　不经意传输协议和轻量级代理不经意传输协议所花费时间的对比

Fig. 10　Comparison of the time spent by oblivious transfer protocol and lightweight proxy oblivio us transfer protocol

从图 10(a)中可以看出，局域网环境中轻量级代理不经意传输协议的时间开销远小于不经意传输。轻量级代理不经意传输协议更注重提高参与方的计算效率，并没有对通信时间进行优化。而在无线局域网环境中，通信在总时间中的占比较大，因此图 10(b)中轻量级代理不经意传输协议花费的时间和不经意传输协议相差并不大。本文将局域网下 Tiny-

Garble花费的时间与本文框架花费的时间进行了对比，结果如表3所列。由于CPU-FPGA在混淆电路生成和计算阶段利用异构计算提高了计算效率，并且提出轻量级代理不经意传输协议减小了不经意传输阶段的计算压力，因此相对于TinyGarble，该框架达到了较高的加速比。文献[22]采用异构计算对Yao协议进行加速，使用的FPGA类型为Stratix V。由于文献[22]的实验并不包含8 bit加法和32 bit汉明距离，因此表3只对比了8 bit乘法和32 bit乘法的计算效率。与文献[22]相比，本框架的8 bit乘法和32 bit乘法的计算效率分别提高了3倍和4倍。文献[22]只加速混淆电路生成阶段，而本框架不仅加速了混淆电路的生成阶段，还加速了混淆电路计算阶段，并且本框架采用了轻量级代理不经意传输协议，降低了不经意传输阶段的计算量。因此，本框架的计算效率比文献[22]方案高。

表3 框架性能评估

Table 3 Evaluation offramework performance

衡量指标	8 bit 加法	32 bit 汉明距离	8 bit 乘法	32 bit 乘法
TinyGarble/ms	23.1	81.2	27.0	107.3
异构安全计算加速框架/us	50.2	211.8	85.7	835.4
与TinyGarble相比的加速比	460	383	315	128
SIFO[22]/us	—	—	293	3 308
与SIFO[22]相比的加速比	—	—	3	4

结束语 本文提出了轻量级异构安全计算加速框架，并利用该框架对安全函数计算进行加速。该框架在Yao协议的基础上添加了代理计算方来提高计算混淆电路的效率，并通过轻量级代理不经意传输协议降低了不经意传输阶段用户的计算压力。与TinyGarble相比，该框架的安全函数计算效率有显著提升。

本文所提的轻量级异构安全计算加速框架能够在安全函数计算中实现较高的计算效率，但是随着应用场景复杂度的提高，该框架的加速效果会下降。因此，今后将围绕如何将该框架更有效地应用于高复杂度的应用场景展开研究。

参 考 文 献

[1] GLAESER E L, NATHANSON C G. An extrapolative model of house price dynamics[J]. Journal of Financial Economics, 2017, 126(1): 147-170.

[2] BOHR A, MEMARZADEH K. The rise of artificial intelligence in healthcare applications[M]// Artificial Intelligence in Healthcare. New York: Academic Press, 2020: 25-60.

[3] LIA B. Globalisation and the future of terrorism: Patterns and predictions[M]. London: Routledge, 2007.

[4] XIA K, LUO Y, XU X, et al. Sgx-fpga: Trusted execution environment for cpu-fpga heterogeneous architecture[C]// 2021 58th ACM/IEEE Design Automation Conference (DAC). NJ: IEEE, 2021: 301-306.

[5] DAUTERMAN E, RATHEE M, POPA R A, et al. Waldo: A private time-series database from function secret sharing[C]// 2022 IEEE Symposium on Security and Privacy (SP). NJ: IEEE, 2022: 2450-2468.

[6] DUAN J, ZHOU J, LI Y, et al. Privacy-preserving and verifiable deep learning inference based on secret sharing[J]. Neurocomputing, 2022, 483: 221-234.

[7] LEE J W, KANG H C, LEE Y, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network[J]. IEEE Access, 2022, 10: 30039-30054.

[8] KNOTT B, VENKATARAMAN S, HANNUN A, et al. Crypten: Secure multi-party computation meets machine learning[J]. Advances in Neural Information Processing Systems, 2021, 34: 4961-4973.

[9] SONGHORI E M, ZEITOUNI S, DESSOUKY G, et al. Garbledcpu: a mips processor for secure computation in hardware[C]// Proceedings of the 53rd Annual Design Automation Conference. NJ: IEEE, 2016: 1-6.

[10] NAOR M, NISSIM K. Communication preserving protocols for secure function evaluation[C]// Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing. NY: ACM, 2001: 590-599.

[11] YAO A C C. How to generate and exchange secrets[C]// 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). NJ: IEEE, 1986: 162-167.

[12] CHOU T, ORLANDI C. The simplest protocol for oblivious transfer[C]// International Conference on Cryptology and Information Security in Latin America. Berlin: Springer, 2015: 40-58.

[13] KHOKHAR A A, PRASANNA V K, SHAABAN M E, et al. Heterogeneous computing: Challenges and opportunities[J]. Computer, 1993, 26(6): 18-27.

[14] MITTAL S, VETTER J S. A survey of CPU-GPU heterogeneous computing techniques[J]. ACM Computing Surveys (CSUR), 2015, 47(4): 1-35.

[15] BRODTKORB A R, DYKEN C, HAGEN T R, et al. State-of-the-art in heterogeneous computing[J]. Scientific Programming, 2010, 18(1): 1-33.

[16] LIU X, OUNIFI H A, GHERBI A, et al. A hybrid GPU-FPGA-based computing platform for machine learning[J]. Procedia Computer Science, 2018, 141: 104-111.

[17] LEESER M, GUNGOR M, HUANG K, et al. Accelerating large garbled circuits on an FPGA-enabled cloud[C]// 2019 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC). NJ: IEEE, 2019: 19-25.

[18] JÄRVINEN K, KOLESNIKOV V, SADEGHI A R, et al. Embedded SFE: Offloading server and network using hardware tokens[C]// International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2010: 207-221.

[19] HUSSAIN S U, ROUHANI B D, GHASEMZADEH M, et al. Maxelerator: FPGA accelerator for privacy preserving multiply-accumulate (MAC) on cloud servers[C]// Proceedings of the 55th Annual Design Automation Conference. NJ: IEEE, 2018: 1-6.

[20] ROUHANI B D, HUSSAIN S U, LAUTER K, et al. ReDCrypt: real-time privacy-preserving deep learning inference in clouds using FPGAs[J]. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 2018, 11(3): 1-21.

[21] FANG X, IOANNIDIS S, LEESER M. Secure function evalua-

tion using an fpga overlay architecture[C]// Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. NY：ACM，2017：257-266.

[22] FANG X，IOANNIDIS S，LEESER M. SIFO：Secure computational infrastructure using FPGA overlays[J]. International Journal of Reconfigurable Computing，2019，2019：1-18.

[23] HUSSAIN S U，KOUSHANFAR F. FASE：FPGA acceleration of secure function evaluation[C]// 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines(FCCM). NJ：IEEE，2019：280-288.

[24] WOLFE P F W. Enabling secure multi-party computation with FPGAs in the datacenter[D]. Boston：Boston University，2021.

[25] HU X，LI M，TIAN J，et al. Efficient Homomorphic Convolution Designs on FPGA for Secure Inference[J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems，2022，30(11)：1691-1704.

[26] SONGHORI E M，HUSSAIN S U，SADEGHI A R，et al. Tinygarble：Highly compressed and scalable sequential garbled circuits[C]// 2015 IEEE Symposium on Security and Privacy. NJ：IEEE，2015：411-428.

[27] KOLESNIKOV V，SCHNEIDER T. Improved garbled circuit：Free XOR gates and applications[C]// International Colloquium on Automata，Languages，and Programming. Berlin：Springer，2008：486-498.

[28] BEAVER D，MICALI S，ROGAWAY P. The round complexity of secure protocols[C]// Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing. NY：ACM，1990：503-513.

ZHAO Chuan，born in 1989，Ph.D，professor，is a member of CCF（No. 73448M）. His main research interests include secure multi-party computing，privacy computing and artificial intelligence security.

ZHAO Shengnan，born in 1994，Ph.D. His main research interests include secure multi-party computing and blockchain technology.

（责任编辑：何杨）