# 项目申请书

向嘉豪

2024 年 6 月 7 日

## 1 立项依据

### 1.1 项目的研究目的、意义

#### 1.1.1 项目的研究目的

随着基础通信设施的不断完善，如 5G 网络的普及，人们对通信的需求逐渐增加。这种需求的增长催生了大量基于通信的应用，如工业互联网、车联网、物联网等。然而，随着这些网络中传输的数据量的增加，安全问题变得越来越严重。这主要是由网络数据中数据包传输机制引起的。幸运的是，对称加密算法提供了一种解决方案，只需双方协商好密钥，就能确保数据在公共信道下的安全传输，而无需调整传输机制。

对称加密算法中，最广泛使用的是 2001 年的 AES[1] 国际算法。它在 WEB、WIFI 等领域，以及服务器与个人计算机上都得到了广泛的应用。然而，为了避免 AES 算法存在的未知门陷，我国在 2006 年提出了 SMS4[2] 对称加密算法来替代 AES 算法。值得一提的是，SMS4 在 2021 年正式成为 ISO 标准并得到了国际认可。这些传统的对称算法在计算资源充足的场景下，能够提供较高的安全性。但在计算资源受限的场景下，由于这些算法的计算复杂度较高，因此需要一种更加轻量级的对称加密算法来满足这种场景的需求。

美国国家标准局 NIST 于 2019 年启动了轻量级密码算法 LWC 的征召。在 56 个轻量级密码算法的竞争中，ASCON[3] 算法经过三轮筛选，最终脱颖而出。到了 2023 年，ASCON 算法已经成为 NIST 的轻量级加密标准。由于 ASCON 算法在资源受限的场景下表现出较高的性能，因此在物联网、车联网等场景中具有广泛的应用前景。

相较而言，我国在轻量级加密算法上的起步较晚，暂时还没有自己的轻量级加密算法标准。然而，在 2019 年由中国密码学会组织的全国密码算法设计竞赛中，一等奖获得者是一种名为 uBlock[4] 的轻量级加密算法。这种算法在计算资源充足的环境下表现优秀，同时在资源受限的场景下，也展现出了高性能。尽管如此，uBlock 算法并未像 ASCON 算法那样得到广泛的学者关注，这使得我国在轻量级加密算法上与国际顶尖水平存在较大的差距。

这种差距不仅体现在轻量级加密算法的设计上，也同样存在于其实现上。这包括硬件和软件实现。具体来说，需要考虑如何高效设计硬件加密的 IP 核，并将其集成入芯片，以实

现硬件级别的安全。同时，也需要考虑如何将加密算法高效实现于 8-bit 或 32-bit 的微控制器中，以确保应用的软件级别安全。我国在这个领域的研究相对较少。因此，本项目的目标是研究轻量级加密算法的实现，以推动我国在轻量级加密算法领域的研究进展。

### 1.1.2 项目的研究意义

在上世纪的算法设计中，对称加密算法的设计主要考虑了安全性。然而，这些设计往往较少考虑其在资源受限的场景下的性能。例如，DES[6] 的实现部分提及其在软件与硬件的实现，但并未给出具体的参考实现方案与实现性能。在 NIST 的 LWC 竞赛中，轻量级加密算法的设计不仅考虑了安全性，还考虑了其在资源受限的场景下的实现性能。这无疑对算法的设计提出了更高的要求。实现算法设计与实现性能之间的桥梁，是本项目研究的出发点。在考虑最新的软硬件平台技术下，将加密算法的组件转化为相应的电路或程序，是本项目研究的手段。更具体来说，本项目的研究意义体现在以下几个方面：

1) 研究轻量级加密算法在专用集成电路 (ASCI) 和现场可编程门阵列 (FPGA) 上的硬件实现。这将提高算法实现的性能，同时确保其硬件级别的安全。2) 研究轻量级加密算法在 8-bit 或 32-bit 的微控制器上的软件实现。这将提高算法实现的性能，同时确保其软件级别的安全。3) 研究轻量级加密算法的软硬件协同实现。在确保算法实现的灵活性的前提下，最大程度地提高算法实现的性能。

## 1.2 国内外研究现状分析和发展趋势

网络数据传输量的增加使得网络安全问题日益严重。对称加密算法是一种能够确保数据在公共信道下安全传输的解决方案。然而，传统的对称加密算法在计算资源受限的场景下计算复杂度较高。因此，需要一种轻量级的对称加密算法来满足这种场景的需求。

在 2011 年之前，轻量级加密算法的研究主要集中在硬件实现上。这是因为硬件实现能够提供更高的性能，如 ISO 的轻量级加密标准 PRESENT[7]。同时，硬件实现也能确保算法的安全性。然而，随着应用场景的增加，性能需求也在变化。现在，不仅关注硬件的电路面积，也开始关注软件加密的执行时间，如美国国家安全局 (NSA)2013 年提出的 SIMON[8]。此外，还关注硬件加密的功耗，如 2015 年亚密会提出的 Midori[9]，以及硬件加密的时延，如 2016 年美密会提出的 SKINNY[10]。因此，轻量级加密算法的内涵正在不断扩展。

### 1.2.1 轻量级加密算法硬件实现

轻量级加密算法的设计和实现都在不断发展和完善。早期的轻量级加密算法主要集中在硬件实现上。例如，Arich 等人对 DES 算法的硬件实现 [11]，他们将 DES 的加密速率提高到了 1 Gbits/s。随着硬件实现的进步，性能指标也从电路面积和加密吞吐量扩展到了功耗和时延等方面。为了实现这些性能指标，提出了一些全新的硬件实现技术。其中，串行、展开、迭代、流水线等技术得到了广泛应用。

串行实现的核心思想是降低加密时的数据带宽。这种方法可以实现对加密组件的重复使用，并减少电路中的寄存器使用数量，以实现更低的电路面积。Good 等人提出了 AES 算法的串行实现 [12]。他们在 Xilinx Spartan-II FPGA (XC2S15) 上实现了面积资源 174 slices 和两个内存块。在这种实现中，最小的数据带宽为一个字节。为了进一步降低数据带宽，Leong 等人提供了 IDEA 算法的比特级数据带宽的串行实现 [13]。然而，对于以 S 盒作为基本加密组件的加密算法，实现一比特级的数据带宽是困难的。为了解决这个问题，Jean 等人为类 SPN 结构的加密算法提出了比特滑动技术 [14]。这种技术可以实现比特级的数据带宽。

展开实现的核心思想是在同一周期内计算加密算法的多轮函数。这种方法可以降低加密所需的延迟，并在引入更多面积的情况下获取更高的吞吐量。Elbirt 等人在 AES 算法竞选的最终轮中使用了展开实现 [15]。他们的 2 轮展开实现 Rijndael 算法实现了最高的吞吐量，这在对比基于轮的迭代实现时尤为明显。Gupta 等人将展开实现与流水线结合 [16]，这极大地提高了 RC4[17] 算法的吞吐量。

迭代实现的核心思想是在同一周期内运算加密算法的一轮函数。通过反馈机制，可以实现多轮函数的计算。这种方法可以减少加密所需的面积，但会增加加密所需的延迟。值得注意的是，迭代实现在面积与吞吐量之间实现了较好的平衡。在 AES 算法竞选的最终轮中，Elbirt 等人的迭代实现的 Rijndael 获得了最高的吞吐量与面积比 [15]。类似的情况也出现在 LWC 竞赛中。ASCON 算法的迭代实现在面积与吞吐量上取得了较好的平衡 [6]。

流水线实现的核心思想是将加密算法的一轮函数分解为多个阶段。这样，每个阶段的计算可以并行进行，实现多个分组同时加密。这种方法可以显著提高加密所需的吞吐量，但会增加实现的面积。Kryjak 等人运用了这种流水线技术 [18]。他们对 CLEFIA[19] 算法进行了实现，极大地提高了加密的吞吐量。然而，考虑到所需的资源，这种实现主要适用于超性能计算场景。

总的来说，对于传统的性能指标，如面积和吞吐量，轻量级加密算法的硬件实现已经有了较好的解决方案。然而，对于新的性能指标，如功耗和时延，还需要进一步的研究。未来的研究重点是如何提出新的硬件实现技术，以适应这些新的性能指标。

### 1.2.2 轻量级加密算法软件实现

轻量级加密算法的软件实现主要集中在微控制器上。这与硬件实现有所不同。微控制器提供了更高的灵活性。例如，可以在不同的应用场景下调整加密算法的参数，或者使用不同的加密算法。同时，微控制器也提供了更高的可移植性。例如，可以在不同的硬件平台上运行加密算法。然而，相比硬件实现，软件实现的性能指标通常较低。

早期的轻量级加密算法的软件实现主要集中在 8-bit 微控制器上。这些微控制器常用于 RFID 和传感器等设备。在 1991 年，Merkle 尝试将 Khufu 算法进行软件实现。这为算法在其他不同场景下的应用打开了新的可能性 [20]。后来，Osvik 将 AES 算法实现在 8-bit 的 AVR 微控制器上，刷新了当时的最快加密记录 [21]。

随着微控制器的发展，32-bit 微控制器成为主流。软件实现的侧重也发生了改变。Rogaway 开始在 32-bit 的现代微控制器上，对 SEAL 算法进行了软件实现 [22]。Bertoni 首次将 AES 算法实现在 32-bit 的微控制器上，并通过多种架构的仿真器对优化实现进行验证 [23]。Schwabe 在结合 ARM 平台在 Cortex-M3 和 M4 上实现了 AES 算法，扩展了 AES 算法的用途 [24]。

在软件实现中，存在一些特殊的情况需要注意。例如，抵抗侧信道攻击通常需要添加额外的操作。如何更高效地实现这些抗攻击的操作，成为了一个重要的研究方向。例如，Rivain 提出了一种高阶掩码技术 [25]。这种技术可以有效地抵抗设备在侧信道攻击中的相关功耗攻击。另一方面，轻量级加密算法在某些场景下具有优势。例如，在区块链等对加密算法吞吐量要求较高的场景下，轻量级加密算法可以发挥重要作用。利用 SIMD 指令集可以提高加密算法的吞吐量 [26]。此外，也可以利用 GPU 来实现加密 [21]。这些都是软件实现的重要方向。

总的来说，软件实现的技术比硬件实现更为丰富。因此，许多研究都集中在如何将加密算法高效地实现在更先进的平台上。未来的研究重点将是开发新的软件实现技术，以适应这些新的平台。同时，也需要考虑如何预防新的攻击方式。

### 1.2.3 轻量级加密算法软硬件协同实现

相比硬件实现和软件实现，软硬件协同实现的研究相对较少。这主要是因为软硬件协同实现需要同时考虑硬件和软件的特性。然而，软硬件协同实现在保证算法性能的同时，也能提高算法的灵活性。例如，当加密算法的标准发生变化时，软硬件协同实现能够更快地适应这些变化。值得注意的是，与硬件实现和软件实现相比，软硬件协同实现起步较晚。

在早期的实现中，加密算法的硬件实现和软件实现通常是分开的。例如，加密算法作为 IP 核心，以片上外设的形式挂载在数据总线上，如 Usselmann 实现的 AES 算法 [27]。由于外设与 CPU 之间的通信开销较大，这种实现方式的性能较低。为了提高性能，一些研究开始尝试将加密算法的硬件实现与软件实现集成在一起。这种实现方式可以减少外设与 CPU 之间的通信开销，从而提高性能。然而，由于各家厂商的 CPU 架构不同且闭源，这种实现方式的研究较少。

开源硬件的发展为软硬件协同实现的研究带来了新的机遇。例如，RISC-V 是一种开源的 CPU 架构，可以自由使用。这种架构的出现，为软硬件协同实现提供了新的研究方向。Marshall 设计了一套轻量级扩展指令集，将 ChaCha 算法实现在 RISC-V 上 [28]。基于扩展指令集，Chen 在 2023 年提出了一组通用扩展指令集在 RISC-V 之上，并将其运用到了 LWC 最终轮的 10 个算法上 [29]。这种方式将加密的硬件电路集成入了 CPU 的执行流水线中，保证了加密操作的速度。同时，加密算法中的通用操作被设计为对应的指令，这样即使加密标准更新，也能保持对加密算法的加速效果。

总的来说，软硬件协同实现为轻量级加密算法的实现开辟了新的研究方向。这种实现方式将不利于软件实现的部分转化为硬件实现，并以此作为基础操作。然后，利用软件指令来

构建整体算法，从而在加密算法的性能与灵活性之间取得平衡。这也为算法设计提供了一条全新的思路。未来的研究重点是将轻量级加密算法中的通用操作设计为扩展指令集或通用加密 IP 核。

## 1.3 参考文献

[1] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." (1999).

[2] Diffie, Whitfield, and George Ledin. "SMS4 encryption algorithm for wireless networks." Cryptology ePrint Archive (2008).

[3] Dobraunig, Christoph, et al. "Ascon v1. 2: Lightweight authenticated encryption and hashing." Journal of Cryptology 34 (2021): 1-42.

[4] Wen-Ling, Wu, et al. "The block cipher uBlock." Journal of Cryptologic Research 6.06 (2019): 690-703.

[5] Pub, F. I. P. S. "Data encryption standard (des)." FIPS PUB (1999): 46-3.

[6] Mohajerani, Kamyar, et al. "FPGA benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process: Methodology, metrics, tools, and results." Cryptology ePrint Archive (2020).

[7] Bogdanov, Andrey, et al. "PRESENT: An ultra-lightweight block cipher." Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9. Springer Berlin Heidelberg, 2007.

[8] Beaulieu, Ray, et al. "The SIMON and SPECK lightweight block ciphers." Proceedings of the 52nd annual design automation conference. 2015.

[9] Banik, Subhadeep, et al. "Midori: A block cipher for low energy." Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. Springer Berlin Heidelberg, 2015.

[10] Beierle, Christof, et al. "The SKINNY family of block ciphers and its low-latency variant MANTIS." Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. Springer Berlin Heidelberg, 2016.

[11] Arich, Touria, and Mohssine Eleuldj. "Hardware implementations of the data encryption standard." The 14th International Conference on Microelectronics,. IEEE, 2002.

[12] Good, Tim, and Mohammed Benaissa. "AES as stream cipher on a small FPGA." 2006 IEEE International Symposium on Circuits and Systems. IEEE, 2006.

[13] Leong, Monk-Ping, et al. "A bit-serial implementation of the international data encryption algorithm IDEA." Proceedings 2000 IEEE Symposium on Field-Programmable

Custom Computing Machines (Cat. No. PR00871). IEEE, 2000.

[14] Jean, Jérémy, et al. "Bit-Sliding: A Generic Technique for Bit-Serial Implementations of SPN-based Primitives: Applications to AES, PRESENT and SKINNY." Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Springer International Publishing, 2017.

[15] Elbirt, Adam J., et al. "An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 9.4 (2001): 545-557.

[16] Gupta, Sourav Sen, et al. "High-performance hardware implementation for RC4 stream cipher." IEEE Transactions on Computers 62.4 (2012): 730-743.

[17] Paul, Goutam, and Subhamoy Maitra. RC4 stream cipher and its variants. CRC press, 2011.

[18] Kryjak, Tomasz, and Marek Gorgon. "Pipeline implementation of the 128-bit block cipher CLEFIA in FPGA." 2009 International Conference on Field Programmable Logic and Applications. IEEE, 2009.

[19] Shirai, Taizo, et al. "The 128-bit blockcipher CLEFIA." Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14. Springer Berlin Heidelberg, 2007.

[20] Merkle, Ralph C. "Fast software encryption functions." Advances in Cryptology-CRYPTO' 90: Proceedings 10. Springer Berlin Heidelberg, 1991.

[21] Osvik, Dag Arne, et al. "Fast software AES encryption." Fast Software Encryption: 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers 17. Springer Berlin Heidelberg, 2010.

[22] Rogaway, Phillip, and Don Coppersmith. "A software-optimized encryption algorithm." Journal of Cryptology 11 (1998): 273-287.

[23] Bertoni, Guido, et al. "Efficient software implementation of AES on 32-bit platforms." Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. Springer Berlin Heidelberg, 2003.

[24] Schwabe, Peter, and Ko Stoffelen. "All the AES you need on Cortex-M3 and M4." International Conference on Selected Areas in Cryptography. Springer International Publishing, 2016.

[25] Rivain, Matthieu, Emmanuel Prouff, and Julien Doget. "Higher-order masking and shuffling for software implementations of block ciphers." International Workshop on Cryptographic Hardware and Embedded Systems. Springer Berlin Heidelberg, 2009.

[26] Xu, Runqing, et al. "High-throughput block cipher implementations with SIMD." Journal of Information Security and Applications 70 (2022): 103333.

[27] Usselmann, Rudolf. "Advanced encryption standard/rijndael ip core." 2007-08-20. https://opencores.org/projects/aes_core (2002).

[28] Marshall, Ben, Daniel Page, and Thinh Hung Pham. "A lightweight ise for chacha on risc-v." 2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP). IEEE, 2021.

[29] Cheng, Hao, et al. "RISC-V instruction set extensions for lightweight symmetric cryptography." IACR Transactions on Cryptographic Hardware and Embedded Systems (2023): 193-237.

## 1.4 项目应用前景和学术价值

项目研究的三个方面，硬件实现、软件实现和软硬件协同实现，都具有广泛的应用前景。硬件实现主要适用于专用集成电路 (ASIC) 和现场可编程门阵列 (FPGA)。在对加密性能和成本有较高要求的场景下，硬件实现具有不可替代的地位。例如，可以应用于 IC 无线射频卡、云加密机等。软件实现主要适用于 8-bit 或 32-bit 的微控制器。在对加密算法部署灵活性有较高要求的场景下，软件实现具有广泛的应用前景。例如，可以应用于智能家居、车联网等。软硬件协同实现主要适用于同时对加密性能、成本和灵活性有较高要求的场景。例如，可以应用于区块链等。

在轻量级加密算法领域，我国相对于国际顶尖水平存在一定的差距。这种差距也体现在轻量级加密算法的实现上。本项目深入研究轻量级加密算法实现的三个方面，包括硬件实现、软件实现和软硬件协同实现。这将有助于推动我国在轻量级加密算法领域的研究进展，弥补与国际水平的差距。同时，这也为我国在轻量级加密算法的标准化过程中提供了实现性能上的评估，推进轻量级加密算法的标准化。

总的来说，本项目的研究成果具有广泛的应用前景和学术价值。对于不同的应用场景，可以提供各种实现方案，以实现更低的成本、更高的性能和更高的灵活性。此外，本项目的研究成果将推动我国在轻量级加密算法领域的研究进展，促进我国在轻量级加密算法实现上的独立自主。

## 1.5 现有研究基础、条件、手段

### 1.5.1 现有研究基础

本项目组主要成员中，有研究生 4 人（包括申请人向嘉豪）。本项目组一直从事轻量级分组密码算法的实现、分析与设计研究，在该领域取得一定的研究成果，目前取得的成果有中科院 3 区论文 1 篇。因此，本项目组对轻量级分组密码方向形成了一定的研究基础，对研究轻量级分组密码算法的实现有着重要意义。

### 1.5.2 现有研究条件

项目组所依托的“嵌入式计算与信息安全研究所”湖南省重点实验室，目前占地面积 80 平米，相应重要的实验研发设备均已购置，能提供项目良好的研究开发和工作环境，学校对科研工作很重视，也能提供足够的科研工作时间。该研究所有一支稳定的队伍（在职在岗的教师 2 人，博士、硕士研究生 30 多人）长期从事轻量级密码算法构造及应用、轻量级密码算法优化、轻量级密码算法安全性分析、密码算法攻击的研究，目前发表多篇 SCI 论文、申请了多项专利，特别在 SoC 芯片、FPGA 实现研究与设计方面积累了宝贵的经验与教训，有着良好的软硬件开发团队作风和项目经验, 有专业资深的老师指导，有产学研项目研究经历，在面向产业化应用研发方面具有较好成果。

### 1.5.3 现有研究手段

本项目组针对目前国内外轻量级分组加密算法优化和安全实现，利用各种多媒体、三大数据库、欧洲密码会议、亚洲密码会议、美国密码会议等渠道持续关注与学习。本项目组针对轻量级分组密码算法的实现优化，结合目前主流硬件平台 FPGA 和软件平台 ARM-Cortex，本项目组目前采用软件与硬件技术优化密码算法实现；针对轻量级密码算法的硬件实现，本项目组有使用迭代、展开、串行、流水线等硬件架构，优化算法的硬件实现，同时结合 S 盒约束条件下小空间搜索和布尔可满足性问题中启发式搜索，对加密算法的组件进行硬件实现优化；在算法的软件实现上，利用 ARMv7-M 中 thumb 指令集，对加密算法进行汇编实现优化；在侧信道分析上，结合深度学习，对优化实现后的算法进行相关功耗分析；在算法实现性能评估上，本项目组在 Xilinx Artix-7 FPGA 以及 ASIC 上进行硬件评估，评估的指标主要为面积、延迟、功耗、吞吐量等。在 8/32 位微处理器平台上进行软件评估，评估指标主要为速度、ROM、RAM 等。

# 2 研究方案

## 2.1 研究目标、研究内容和拟解决的关键问题

### 2.1.1 研究目标

在轻量级分组密码算法设计上，设计者会考虑算法的安全、性能、成本等因素。但是设计者对算法实现存在局限性，多数算法实现未能达到最优。对于将算法应用到具体场景下，研究轻量级分组密码算法的优化实现有着重要意义。因此，本项目的研究目标是研究轻量级分组密码算法的多场景优化实现，具体来说有以下三方面的目标：

1）研究轻量级分组密码算法在专用集成电路 (ASIC) 和现场可编程门阵列 (FPGA) 上的硬件实现。这将提高算法实现的性能，同时确保其硬件级别的安全。2）研究轻量级分组密码算法在 8-bit 或 32-bit 的微控制器上的软件实现。这将提高算法实现的性能，同时确保

其软件级别的安全。3）研究轻量级分组密码算法的软硬件协同实现。在确保算法实现的灵活性的前提下，最大程度地提高算法实现的性能。

### 2.1.2 研究内容

本研究项目的内容是研究轻量级分组密码算法的多场景优化实现。研究轻量级分组密码在硬件、软件和软硬件平台下优化实现的关键技术，具体来说，包括以下三个方面的内容：

1）研究轻量级分组密码算法在专用集成电路 (ASIC) 和现场可编程门阵列 (FPGA) 上的硬件实现。

在需要高性能的场景下，硬件实现是不可替代的。其中就加密算法部署的规模，ASIC 是一种专用集成电路，可以提供更高的性能，在大批量部署下具有优势。同时 FPGA 是一种现场可编程门阵列，在少批量部署上，可以提供更高的灵活性。本项目将研究轻量级分组密码算法在 ASIC 和 FPGA 上的硬件实现，以提高算法实现的性能，同时考虑硬件实现过程中泄露的侧信道信息，防止功耗、故障攻击等，为算法提供硬件实现安全。

2）研究轻量级分组密码算法在微控制器上的软件实现。

在物联网、智能家居等场景下，终端设备采用资源受限的微控制器，将硬件实现的加密算法部署到微控制器上，需要片外加密芯片或片内加密模块，这样会增加成本。因此，软件实现是一种更为经济的解决方案。本项目将研究轻量级分组密码算法在 8-bit 或 32-bit 的微控制器上的软件实现，以提高算法实现的性能，同时考虑软件实现过程中泄露的侧信道信息，防止时间、缓存攻击等，为算法提供软件实现安全。

3）研究轻量级分组密码算法的软硬件协同实现。

在一些场景下，需要兼顾硬件实现的性能和软件实现的灵活性。例如，区块链等场景下，需要高性能的加密算法，同时也需要灵活的加密算法。本项目将研究轻量级分组密码算法的软硬件协同实现，将加密算法的通用操作设计为扩展指令集或通用加密 IP 核，以提高算法的性能和灵活性。

### 2.1.3 拟解决的关键问题

1）研究轻量级分组密码组件与结构对于硬件实现的影响。在硬件实现原语之上，优化实现轻量级分组密码，同时设计硬件实现的侧信道防护方案。

2）研究轻量级分组密码算法微控制器上的软件实现。在微控器的基础指令集与扩展指令集上，优化实现轻量级分组密码，同时设计软件实现的侧信道防护方案。

3）研究轻量级分组密码算法的软硬件协同实现。利用硬件优化技术设计高效扩展指令集或通用加密 IP 核，结合软件优化技术提高算法部署的性能和灵活性，同时给出实现的侧信道防护方案。

## 2.2 拟采取的研究方法及可行性分析

### 2.2.1 拟采取的研究方法

鉴于本项目的研究内容以及项目组在相关领域的工作积累，拟采取如下详细研究方案和技术路线:

1) 研究轻量级分组密码算法的硬件实现。

在硬件实现方面，本项目将研究轻量级分组密码算法的硬件实现技术，从算法架构、组件优化和安全实现出发，确保算法实现的性能与硬件安全。在算法架构层，结合轻量级分组密码的特点，优化实现轻量级分组密码。举例来说，将 CRAFT 算法进行串行架构实现，分析调整加密组件的执行次序，同时对加密组件进行时序状态分析，确保算法高效运行，最终减少数据带宽，降低硬件实现面积。在组件优化层，对轻量级分组密码的 S 盒、P 盒等组件进行优化，提高组件的性能。举例来说，将 CRAFT 算法的 S 盒在布尔满足性（SAT）上进行建模，约束 S 盒中的电路门数，求解出低面积 S 盒硬件实现。在硬件安全上，对于故障攻击，通过冗余检测电路和预计算编码表确保安全，在功耗攻击上，通过随机运算电路和掩码技术确保安全。

2) 研究轻量级分组密码算法在微控制器上的软件实现。

在软件实现方面，从软件运行的平台、加密算法组件优化和安全实现出发，确保算法实现的性能与软件安全。在软件运行的平台层，本项目将研究轻量级分组密码算法在主流 32-bit 的微控制器上的实现，包括 ARM-Cortex M3、M4, RISC-V 等。结合精简指令集平台所具备的基本指令，对算法进行汇编优化，同时探究特殊平台下的指令对算法的影响，如 ARM-Cortex M4 中循环移位指令对算法的加速效果。在加密算法组件优化层，对轻量级分组密码的 S 盒、P 盒和扩散矩阵等组件进行优化，提高组件的性能。举例来说，将 CRAFT 算法的 S 盒在可满足性（SMT）上比特切片建模，通过限制模型的指令数，提高 S 盒的执行效率。在软件安全上，对于时间攻击，通过随机延迟确保安全，在缓存攻击上，通过比特切片减少算法内存使用，保证组件相同运行时间，减少侧信道信息泄露。

3）研究轻量级分组密码算法的软硬件协同实现。

在软硬件协同实现上，本项目将研究轻量级分组密码算法的软硬件协同实现技术，在开源的 RISC-V 平台上设计高效扩展指令集及通用加密外设。在扩展指令集上，结合寄存器的数量限制以及指令的设计准则，将加密算法中的通用操作设计为扩展指令，例如，加密算法中起到扩散功能的线性层，设计比特级置换指令，对起到混淆功能的非线性层，设计可配置的替换指令。在通用加密外设上，可以在不更改芯片核心流水线的情况下，将通用加密硬件电路集成到芯片中，将加密算法的通用操作设计为外设访问，通过配置外设的配置寄存器，实现不同加密算法的加速。同时，设计软件接口，将扩展指令集和通用加密外设的操作封装为软件指令，提供给软件开发者使用。

### 2.2.2 可行性分析

首先，针对本项目的主要研究内容和研究目标，项目组做了充分的准备并制定了详细的研究方案和细致的研究计划。同时，针对本项目的各项主要研究内容，本项目组在理论和技术方面均已具备大量的研究基础和工作积累，这将会积极促进本项目的顺利的启动和研究深入。其次，在研究内容的描述中，我们尽可能细化，明确研究对象，具体研究方法。尤其是在研究方案的阐述中，我们明确了需解决的问题，初步拟定了采取的技术方法等，研究计划清楚且现实可行。最后，项目的研究目标和预期成果制定合理适度，能够保证研究的广度和深度。综上所述，我们认为完成本项目的研究是切实可行的。

## 2.3 本项目的创新之处

本项目的创新之处主要体现在以下三个方面：1）本项目将研究轻量级分组密码算法的侧信道攻击防护。在硬件实现和软件实现中，设计防护方案，防止功耗、时间、缓存等侧信道攻击，保证算法实现的安全性。2）本项目将研究轻量级分组密码算法的软硬件协同实现。将加密算法的通用操作设计为扩展指令集或通用加密 IP 核，以提高算法的性能和灵活性，面向未来多加密算法场景。3）目前对于轻量级分组密码算法的实现，主要集中在单一实现平台上。本项目将研究轻量级分组密码算法的多环境平台下的实现，为设计多平台下高效密码算法设计提供研究基础。

## 2.4 预期研究进展

1）2024 年 6 月—2024 年 11 月确定总体方案，对项目实施做出具体安排：文献、资料的跟踪及收集，研究和学习轻量级密码算法硬件，软件以及软硬件协同优化实现，为本项目算法设计打下基础。

2）2024 年 12 月—2025 年 6 月研究轻量级分组密码的硬件优化实现，并应用到对具体场景。组件上，对轻量级 S 盒在不同工艺库下的最优实现进行研究，架构上，分析已有架构对轻量级分组密码实现的影响。

3）2025 年 7 月—2025 年 11 月研究轻量级分组密码的软件优化实现，并在嵌入设备中对其验证。组件上，对轻量级 S 盒最优比特切片实现进行研究，平台上，研究轻量级分组密码在嵌入式平台实现的难点。

4）2025 年 12 月—2026 年 6 月通过硬件技术设计出通用加密组件，结合软件实现优化技术，提出一种软硬件协同，优化实现轻量级分组密码的方案。

对项目研究工作进行总结，准备项目结题，撰写总结报告，参加结题答辩。