# 专利改

👌 收到，你的意思是：在专利里聚焦两个最核心的点，**主保护点** **=** **SumIR 生成机制**，**次保护点** **=** **IR 翻译无缝接入**。  
这样会让权利要求更聚焦、保护范围更清晰，也方便以后维权。下面我给你写一个可以直接放进说明书的“技术方案 + 有益效果 + 取证手段”三段式。

# 技术方案（聚焦核心保护点）

本发明提出一种面向鸿蒙 Node-API 的跨语言静态分析方法，其核心在于：

1. **SumIR 摘要生成（主保护点）**  
   针对原生 C/C++ 模块的实现逻辑，设计一种最小完备的三指令抽象 **SumIR**，包括：

* + **CALL 指令**：用于描述 Node-API 调用及外部函数调用的参数—返回依赖；

* + **PHI 指令**：用于表示控制流分支的合流依赖；

* + **RET 指令**：用于表示函数返回语义。  
    通过静态分析原生函数的数据依赖关系，将复杂逻辑压缩为由上述三类指令构成的摘要，得到独立的 **SumIR 表示**。

1. **IR 翻译与无缝接入（次保护点）**  
   基于 SumIR 摘要，在 ArkTS 层进行翻译：

* + 在 ArkIR 调用点识别对应的原生函数；

* + 将 SumIR 指令逐一映射为 ArkIR 的等价语句；

* + 自动构建出合成函数体，并与调用点建立调用边。  
    经此步骤，原生逻辑被透明地嵌入到 ArkIR 控制流中，原本缺失的跨语言数据流可以直接被下游分析器识别。

# 有益效果

* **解决鸿蒙缺口**：首次针对 Node-API 跨语言机制提出专门的静态分析方法，弥补现有方法的空白。

* **抽象能力强**：SumIR 以三指令覆盖参数传递、分支合流和返回，保证了简洁性与完备性。

* **无缝衔接**：翻译得到的合成 ArkIR 函数体可直接与调用点相连，不需要改造下游分析工具。

* **跨语言可见性**：原本在 ArkTS 中不可见的原生数据流路径被完整暴露，从而提升敏感信息检测、隐私合规审计等分析能力。

# 取证手段

为验证和界定本发明的保护点，可采用以下取证方式：

1. **SumIR 摘要验证（主保护点对应）**

* + 输出文件中应存在由 CALL、PHI、RET 三类指令构成的指令序列；

* + 检查 CALL 指令是否涵盖参数依赖，PHI 指令是否建模控制流合流，RET 指令是否体现返回值；

* + 同一原生函数经本方法处理，均应生成可复现的 SumIR 表示。

1. **IR 接入验证（次保护点对应）**

* + 在 ArkIR 中应出现与 SumIR 对应的合成函数体；

* + 检查合成函数体与调用点之间是否建立了新增调用边；

* + 对比“接入前后”的 ArkIR 控制流图，可验证跨语言数据流的显性化；

* + 运行现有数据流分析器时，应能够追踪到跨越 ArkTS 与原生模块的敏感路径。

以上证据（SumIR 文件、合成函数体、调用边、分析结果）均可保存和比对，作为本发明技术特征的客观证明。

👉 这样写，就把 **SumIR（主保护点）** 和 **无缝接入（次保护点）** 对应的“方案—效果—取证”完整闭环地呈现出来了。

要不要我帮你再把这一套表述，进一步改写成 **权利要求书条款** 的形式？这样你直接能放到专利申请里。

# deepseek

好的，明白你的需求。专利申请的核心在于清晰地定义“发明点”（即你独创的技术方案）、阐述其与现有技术的“区别点”、并说明如何“实施”和“取证”。

结合你的论文和导师意见，我将帮你聚焦于 \*\*“SumIR生成”\*\* 和 \*\*“IR翻译与无缝接入”\*\* 这两个最核心的环节，撰写一份符合专利要求的、突出技术独特性和保护点的文字说明。

### \*\*核心技术方案与保护点\*\*

本发明提供了一种面向HarmonyOS应用的跨语言静态分析方法，其核心在于通过一种名为\*\*SumIR（Summary Intermediate Representation）\*\* 的中间表示，对原生（C/C++）代码中的跨语言语义进行摘要式建模，并将其无缝翻译至鸿蒙生态标准中间表示\*\*ArkIR\*\*，使得现有仅支持ArkTS的分析工具无需改动即可具备跨语言分析能力。

本发明的核心保护点可分为以下两个主要方面：

#### \*\*一、 基于SumIR的跨语言数据流语义摘要生成方法\*\*

\*\*1. 技术问题：\*\*  
Node-API采用动态描述符注册和上下文句柄（如napi\_env, napi\_callback\_info）传递参数，其数据流（如参数如何从ArkTS端被提取、在原生端如何加工、最终传向何处）在二进制或源代码中均是间接且隐式的，对静态分析极不友好。

\*\*2. 解决方案（独创性流程）：\*\*  
本发明提出了一种从原生代码中提取关键语义并生成SumIR的方法，其流程如下图所示，主要包括语义提取与指令化封装两个阶段：

\*\*3. 保护点与区分点：\*\*

* \*\*保护点1：\*\* 一种基于静态分析（不限于二进制或源码），从使用Node-API的原生代码中生成SumIR的方法，其特征在于，所述SumIR由CALL、PHI、RET三类指令构成，\*\*专注于抽象跨语言数据流而无需还原完整程序状态\*\*。

* \*\*保护点2：\*\* 根据权利要求1所述的方法，其特征在于，所述CALL指令用于建模\*\*Node-API函数（如napi\_get\_cb\_info）及安全敏感函数（如系统日志函数）的调用及其输入输出关系\*\*。

* \*\*保护点3：\*\* 根据权利要求1所述的方法，其特征在于，所述PHI指令用于\*\*显式表征控制流合流点的数据汇入情况\*\*，以支持路径敏感的污点分析。

* \*\*区分点：\*\* 与现有Android跨语言分析技术（如基于JNI函数名映射、基于JNIEnv参数直接传递）不同，本发明是\*\*首个针对Node-API“描述符注册+上下文句柄”这一独特交互模型\*\*进行语义摘要的方案。

\*\*4. 取证手段：\*\*  
检查目标系统是否对HarmonyOS原生库进行静态分析，并生成了一种中间表示（SumIR），该中间表示\*\*是否包含以CALL、PHI、RET为核心的指令集\*\*，并以此抽象了参数提取、数据转换、敏感函数调用等行为。

#### \*\*二、 SumIR至ArkIR的语义保持翻译与无缝生态集成方法\*\*

\*\*1. 技术问题：\*\*  
生成的SumIR是一种与平台无关的摘要，无法被鸿蒙现有分析工具（如ArkAnalyzer）直接理解。如何将其转换为ArkIR并融入现有分析流程，是实现“无缝接入”的关键。

\*\*2. 解决方案（独创性流程）：\*\*  
本发明提出了一种将SumIR翻译为ArkIR函数体并重建跨语言调用图的机制，其核心流程如下图所示：

\*\*3. 保护点与区分点：\*\*

* \*\*保护点4：\*\* 一种将SumIR翻译为ArkIR兼容代码的方法，其特征在于，包括以下步骤：\*\*基于ArkTS调用点上下文进行类型推导，将SumIR中的抽象句柄解析为具体ArkTS类型\*\*；将SumIR指令\*\*一对一映射为语义等价的ArkIR操作\*\*。

* \*\*保护点5：\*\* 一种跨语言调用图重建方法，其特征在于，通过\*\*为原生函数创建包含翻译后ArkIR指令的函数体，并替换原有调用桩（stub），从而在调用图中建立完整的跨语言调用边\*\*。

* \*\*保护点6：\*\* 一种兼容现有HarmonyOS静态分析生态的系统，其特征在于，\*\*下游分析工具无需任何修改\*\*，即可像分析普通ArkTS函数一样，对翻译后的、包含原生语义的ArkIR函数体进行数据流、控制流等分析。

* \*\*区分点：\*\* 本方案的“无缝集成”特性是独特的。现有安卓方案（如JuCify）需修改分析器或中间表示以支持JNI，而本发明\*\*通过“前端摘要生成+后端IR翻译”的架构，使分析器无需感知分析的是原生代码\*\*，极大降低了生态集成成本。

\*\*4. 取证手段：\*\*  
检查目标系统是否将分析原生代码后得到的摘要信息，\*\*转换为与分析工具链原有中间表示（ArkIR）完全兼容的代码段\*\*；检查分析工具\*\*是否在不修改自身的情况下\*\*，成功识别出经由该代码段传递的、跨越ArkTS与原生边界的数据流。

### \*\*总结陈述\*\*

本发明通过独创的\*\*SumIR摘要生成\*\*和\*\*语义保持的IR翻译\*\*这两大核心技术，首次实现了对HarmonyOS Node-API跨语言交互的精准、有效的静态分析，并创造性地解决了与现有生态的\*\*无缝集成问题\*\*。其核心创新不在于静态分析算法本身，而在于\*\*针对HarmonyOS独特架构所设计的“语义抽象-表示-翻译”的全新范式\*\*。这为在HarmonyOS生态中进行安全漏洞挖掘、隐私合规检测等提供了前所未有的跨语言分析能力。

希望这份详细的梳理能帮助你更好地撰写专利正文。如果需要我将此内容转化为更正式的专利说明书句式或权利要求书草案，我可以继续协助。

## 技术设计

## 未命名

非常好，这个版本非常精炼，切中要害，非常适合专利撰写。我们在此基础上，进一步用更规范、更具保护性的专利语言进行深化和扩展，形成可以直接使用的说明书核心部分和权利要求草案。

### \*\*说明书核心部分 (用于专利具体实施方式)\*\*

#### \*\*一、 技术领域\*\*

本发明涉及计算机软件安全静态分析领域，特别是一种针对华为鸿蒙（HarmonyOS）操作系统中，ArkTS语言与C/C++原生代码通过Node-API进行跨语言交互的静态分析方法。

#### \*\*二、 背景技术\*\*

随着移动应用功能的复杂化，跨语言编程成为提升性能和控制硬件的关键手段。在Android系统中，Java与C/C++通过Java Native Interface（JNI）交互，并已涌现出如JN-SAF、JuCify等多种跨语言静态分析工具。

然而，鸿蒙操作系统采用了截然不同的跨语言机制。其首选应用开发语言ArkTS通过Node-API与原生模块交互。Node-API采用\*\*动态描述符注册\*\*和\*\*不透明的上下文句柄（如napi\_env, napi\_callback\_info）\*\* 来传递参数和上下文，这与JNI的静态函数映射和显式参数传递有本质区别。现有基于JNI模型的安卓分析技术无法直接应用于鸿蒙生态，导致针对HarmonyOS应用的静态分析工具存在\*\*跨语言数据流不可见\*\*的重大盲区，使得隐私泄露、恶意行为等安全风险无法被有效检测。

因此，迫切需要一种专为HarmonyOS Node-API架构设计的跨语言静态分析方案。

#### \*\*三、 发明内容\*\*

##### \*\*（一） 要解决的技术问题\*\*

本发明旨在解决现有技术无法对HarmonyOS应用中ArkTS与C/C++之间的跨语言数据流进行有效静态分析的问题，克服Node-API的动态性和不透明性所带来的分析障碍。

##### \*\*（二） 核心技术方案\*\*

本发明的整体思路是：通过一种前端摘要生成机制，将原生代码的复杂语义抽象为一个轻量的、标准化的中间表示（SumIR），再通过一种后端翻译机制，将该中间表示无缝转换为鸿蒙现有分析工具链可识别的形式（ArkIR），从而实现对跨语言数据流的透明分析。

参考下图，其核心流程包括：

具体而言，本发明提供以下核心方法：

\*\*1. 一种面向Node-API的跨语言语义摘要生成方法（对应主保护点）\*\*  
该方法包括：

* 对HarmonyOS应用中的原生C/C++模块进行静态分析，提取其内部的数据流与控制流；

* 将分析所得的关键语义信息，抽象为一个由三类指令构成的摘要式中间表示（SumIR）：

* + \*\*CALL指令\*\*：用于抽象表示对Node-API函数（如napi\_get\_cb\_info）及所有外部安全敏感函数（如系统日志、网络IO函数）的调用，并刻画其输入参数与输出结果之间的依赖关系；

* + \*\*PHI指令\*\*：用于抽象表示程序控制流中不同分支路径的合流点，确保路径敏感的污点分析得以进行；

* + \*\*RET指令\*\*：用于抽象表示函数的返回值语义，指明返回值的来源。

* 通过上述三类指令的组合，完整刻画原生函数的行为摘要，而无需还原其全部程序状态。

\*\*2. 一种SumIR至ArkIR的翻译与无缝生态集成方法（对应次保护点）\*\*  
该方法包括：

* \*\*调用点识别\*\*：在ArkTS代码中定位调用上述原生函数的调用点；

* \*\*类型推导\*\*：根据ArkTS调用点的上下文信息，推断出SumIR中抽象值在ArkTS中的具体数据类型；

* \*\*指令翻译\*\*：将SumIR中的每一条指令一对一地映射为语义完全等价的ArkIR操作序列；

* + 将CALL指令翻译为ArkIR函数调用语句；

* + 将PHI指令翻译为ArkIR的条件赋值语句；

* + 将RET指令翻译为ArkIR的返回语句。

* \*\*控制流重建\*\*：将翻译得到的ArkIR操作序列集成为一个完整的、合成的ArkIR函数体，并将该函数体与ArkTS代码中的调用点进行连接，在全局调用图中建立完整的跨语言调用边，使得下游分析器能够透明地处理跨语言数据流。

##### \*\*（三） 有益效果\*\*

与现有技术相比，本发明具有以下显著优点：

1. \*\*首创性\*\*：填补了HarmonyOS生态中跨语言静态分析工具的空白，是首个针对Node-API机制设计的静态分析方案。

1. \*\*抽象精准\*\*：提出的SumIR中间表示以最小化的三类指令（CALL/PHI/RET），实现了对Node-API复杂交互语义的完备抽象，兼顾了简洁性与分析精度。

1. \*\*生态无缝集成\*\*：通过将SumIR翻译为标准ArkIR，使得本发明生成的摘要能够被鸿蒙现有静态分析工具（如ArkAnalyzer）直接、无缝地利用，\*\*无需对下游分析器做任何修改\*\*，极大降低了部署成本。

1. \*\*提升安全性\*\*：使得原本不可见的跨语言隐私数据流、违规操作等安全威胁变得可见，显著增强了HarmonyOS应用的安全检测能力。

### \*\*四、 取证与验证手段\*\*

为验证本专利的实施，可采用以下证据链：

1. \*\*SumIR生成验证\*\*：检查分析系统的输出中是否包含扩展名为.sumir或类似格式的文件，该文件内容应由一系列符合预定规范的CALL、PHI、RET指令构成，并能准确反映原生函数对Node-API的调用序列、数据加工过程和返回值逻辑。

1. \*\*IR翻译验证\*\*：检查静态分析工具（如ArkAnalyzer）生成的中间表示（ArkIR）中，是否存在与原生函数同名的、\*\*新生成的合成函数体\*\*。该函数体应由ArkIR指令构成，且其语义与前述SumIR摘要完全一致。

1. \*\*调用图验证\*\*：通过对比集成前后的调用图，可证实ArkTS调用点与上述合成函数体之间已建立了新的、直接的调用边（Call Edge）。

1. \*\*分析结果验证\*\*：运行未经修改的下游污点分析工具，若该工具能成功报告一条从ArkTS源点出发、穿过该合成函数体、最终到达原生代码中 sink点 的完整数据流路径，则为本发明有效性的直接证明。

### \*\*五、 权利要求书（草案）\*\*

1. 一种用于HarmonyOS应用的跨语言静态分析方法，其特征在于，包括：  
   步骤一：通过对应用中的原生C/C++代码进行静态分析，生成一种摘要中间表示（SumIR），其中所述SumIR由CALL、PHI、RET三类指令组成，用于抽象表示所述原生代码中的跨语言数据流语义；  
   步骤二：将所述SumIR翻译为鸿蒙标准中间表示ArkIR，生成对应的合成函数体，并将该函数体集成至应用的全局调用图中，使下游静态分析工具能够识别跨语言数据流。

1. 根据权利要求1所述的方法，其特征在于，所述步骤一中，生成SumIR包括：

* + 使用CALL指令来抽象表示对Node-API函数或外部安全敏感函数的调用及输入输出关系；

* + 使用PHI指令来抽象表示程序控制流中不同分支路径的合流点；

* + 使用RET指令来抽象表示函数的返回值语义。

1. 根据权利要求1所述的方法，其特征在于，所述步骤二中，将SumIR翻译为ArkIR包括：

* + 识别ArkTS代码中调用原生函数的调用点；

* + 根据调用点上下文推断SumIR中抽象值的具体ArkTS类型；

* + 将SumIR中的CALL、PHI、RET指令分别映射为语义等价的ArkIR函数调用、条件赋值和返回语句。

1. 根据权利要求1所述的方法，其特征在于，所述步骤二中，集成所述合成函数体后，\*\*无需修改下游静态分析工具\*\*，该工具即能对跨语言数据流进行分析。

1. 一种计算机系统，其特征在于，包括存储器和处理器，所述存储器存储有计算机程序，所述处理器执行所述程序时实现如权利要求1-4中任一所述方法的步骤。

1. 一种计算机可读存储介质，其上存储有计算机指令，其特征在于，所述指令被处理器执行时实现如权利要求1-4中任一所述方法的步骤。

希望这份更加详实和规范的版本能满足你的需求。如需进一步调整，请随时提出。

# 未命名

太好了，我把你给的论文 PDF 和“创新点说明”初稿都看过了，结合导师的 7 条点评，给你一份“对照—取材—落地”的修改路线图。下面每条都会标清：该删/该补/从哪段现有材料抽取（用引用标注）。

# 总体方向（一句话）

把故事线从“完整流程+工具清单”切换到“一个可落地的跨语言保护点”：**把原生代码的数据流抽象为 SumIR 指令（CALL/PHI/RET），并在 ArkTS 侧自动重建函数体与调用边，实现与现有分析生态无缝衔接**。其余背景、流程细节与工具名一律收紧或删去。

## 按导师意见逐条的修改方向

### 1）“跨语言技术背景写一页就可以了”

* **保留**：鸿蒙跨语言采用 Node-API、与 Android JNI 架构差异点（描述符式注册、napi 句柄/间接参数获取导致静态分析困难）。这些点在论文“Background & Motivation”已写清，可压缩成一页。

* **做法**：一段讲 Node-API 绑定与参数提取（图 3 的两种绑定+参数获取模式做口述化），一段讲“现有单语言分析看不到原生语义”，以动机例子作证据。

### 2）“要写暂无鸿蒙跨语言加护，并例举安卓跨语言技术”

* **表达建议**：在背景末尾明确写出：**鸿蒙生态尚缺乏面向 Node-API 的跨语言静态分析/防护能力**（可称“跨语言分析与防护能力空白”），**Android 侧已有一批方法但不可直接迁移**（举 JN-SAF、JuCify、NativeSummary 三个代表即可）。这些在论文相关工作里都有论述与定位。

* **注意**：“加护”建议改写为“分析与防护能力/机制”，避免歧义。

### 3）“没有写出保护点具体是什么，要详细介绍数据流抽象为指令以及翻译到和后续分析无缝衔接的具体过程”

* **主保护点**（写到权利要求+发明内容）：

* 1. **Node-API 绑定解析**：从二进制/源码解析 ArkTS 可见名 ↔ 原生函数实现的绑定条目；

* 1. **SumIR 摘要**：以 **CALL/PHI/RET** 三类指令记录“参数提取—外部调用—返回”的数据流关系（支持多返回/指针出参/字符串与缓冲区等）；

* 1. **类型回填与翻译**：在 ArkTS 调用点做 **类型推断**，把 SumIR 系统化翻译为 **ArkIR 函数体**；

* 1. **调用边连接与 CFG 融合**：把 ArkTS 调用点与重建函数体连边，纳入既有分析器，无需改造。

* **可直接取材**：

* + SumIR 三指令定义与设计原则；翻译流水线四阶段（“调用点识别→类型推断→指令映射→CFG 集成”）。

* + 初稿已有“SumIR 三指令”与“完整调用图构建”的表述，把它们上收为**保护点技术特征**而非实现段落。

### 4）“不用详细介绍流程，预处理细节、端到端细节不用详细介绍”

* **删/收**：初稿中的“预处理/Bitcode 抽取/项目级合并/逆向套件”等实现流水线请合并为一句“支持二进制或源码输入，统一生成 SumIR”。不要展开工具与步骤。

* **在论文取材**：把“Implementation/Evaluation”的细节挪到**实施例**，最多 2–3 段作为可选实现。

### 5）“二进制和源代码两层级可以弱化”

* **写法**：一句话交代“双路径输入（ELF/LLVM Bitcode），**输出统一 SumIR 与同一翻译管线**”。不要再分别画两大模块。

### 6）“不必要列出三方工具，图也改”

* **统一删/替换**：删 Ghidra、SVF、WLLVM、BinAbsInspector、ArkAnalyzer、Tool-B/Tool-F 等**第三方或内测工具名**；图中用“Binary Analyzer / Source Analyzer / IR Translator”抽象块替换。

* **图意不变**：**输入→SumIR→ArkIR函数体→连接调用边** 的主链必须保留，其它工具图标与箭头去掉。

### 7）“保护点的表述需要优化”

* **从“做了什么”改为“如何做+限定点”**：把“摘要翻译的生态适配架构”“双层级支持”等宽泛措辞，改为**可检索、可比对的技术特征要点**（见下方“权利要求模板”）。

## 建议的文档/PPT重构

### A. 专利文本骨架（建议 5 章）

1. **技术背景（≤1页）**：Node-API 跨语言特点与分析难点；Android 已有方法但不可迁移；鸿蒙缺口。

1. **发明目的**：在不改动上游/下游分析器的前提下补齐鸿蒙跨语言数据流。

1. **技术方案（核心）**：四步法（绑定解析→SumIR→类型回填翻译→调用边连接），逐条写**结构/流程特征**。

1. **有益效果**：使原先“原生语义不可见”的跨语言流可被既有污点分析识别（用图示意，不提第三方工具）。

1. **实施例**：给二进制或源码任一条路径的小例子（含典型 Node-API 如 napi\_get\_cb\_info、字符串提取与日志泄露链示意）。

### B. PPT 结构（6–8 页）

* 背景 1 页 → 问题 1 页 → **核心发明点 2–3 页（SumIR+翻译+连边）** → 对比“有/无本发明”的检测效果 1 页 → 适配生态/落地 1 页。

* **一张主图**：只保留“ArkTS 调用点 ↔（重建函数体）↔ 原生摘要”的通路。

## “保护点”精炼表述（可直接改写到权利要求/发明内容）

**独立权利要求（建议范式）**  
一种面向鸿蒙 Node-API 的跨语言静态分析方法，包括：  
a) 基于对原生模块的绑定解析，确定 ArkTS 可见函数标识与原生实现的对应关系；  
b) 对原生实现执行静态分析，生成以 **CALL/PHI/RET** 为指令集的**摘要中间表示 SumIR**，所述 CALL 指令表征 Node-API/外部调用的输入-输出依赖，所述 PHI 指令表征路径合流的数据依赖，所述 RET 指令表征返回语义；  
c) 在 ArkTS 调用点处根据函数签名进行**类型推断**，将 SumIR **翻译为 ArkIR 函数体**；  
d) 将所述 ArkIR 函数体**插入控制流图**并与 ArkTS 调用点**建立调用边**，使下游分析无需改动即可覆盖跨语言数据流。

**从属要点（择优挑几条写）**

* 绑定解析同时支持二进制与源码输入，输出统一 SumIR；

* CALL 指令支持多返回与指针出参的**多输出**建模；

* 翻译阶段的类型推断将 napi\_value Opaque 句柄回填为 ArkTS 具体类型；

* 插入的函数体与调用边不改变 ArkTS 源代码与构建流程，仅在 IR 层集成。

## 需要删除/弱化的清单（对应你现稿中的位置）

* **工具名与实现细节**：WLLVM/SVF/Ghidra/BinAbsInspector、Tool-B/Tool-F、ArkAnalyzer 等全部删或以“二进制/源码分析器、IR 翻译器”泛称替代（专利里不必露出三方名）。

* **预处理/端到端流水线**：初稿“预处理、Bitcode 合并、逆向套件、项目级合并”等整段压缩为一句。

* **双层级展开**：把“源代码级/二进制级”两大分节合并为“输入形态不同→SumIR 相同”。

## 图纸与文字的小技巧

* **图**：用 4 个盒子即可：Native Module →（绑定解析+原生分析）→ **SumIR** →（类型推断+翻译）→ **ArkIR 函数体** →（连边）→ 下游分析。

* **术语统一**：SumIR 指令集（CALL/PHI/RET）、类型推断、ArkIR 函数体重建、调用边连接，这些词在全文和图中保持一致。

## 你可以马上做的 5 个动作

1. **删**背景多余页与工具清单；

1. **补**“鸿蒙缺口+Android 例举”一段落（列 2–3 篇代表即可）；

1. **上收**SumIR/翻译/连边为“保护点”章节，并按上面的独立+从属要点落到权利要求；

1. **改图**为“SumIR→ArkIR→连边”的一张主图；

1. **实施例**放一个“获取敏感→原生日志泄露→被污点分析识别”的例子（文字+小示意）。

如果你愿意，我可以基于这份路线图，直接把\*\*专利“技术方案/保护点/有益效果/附图说明”\*\*四个核心段落改写成定稿版中文；也可以按上面的 PPT 架构把演示稿精简成 8 页以内。

# 未命名

好的 👍 我理解了，你现在需要的是一份专利申请文本里“技术方案”层面的核心文字，既要写出流程细节（方便说明差异性和创新性），又要把“保护点”明确提炼出来，尤其聚焦在 **SumIR 生成** 和 **IR 翻译无缝接入** 这两块。下面我给你一个结构化的版本，分为：

1. **技术流程细节（含配图思路）**

1. **核心保护点（可写入权利要求）**

1. **取证手段与效果（支撑专利的有益效果）**

## 一、技术流程细节

（配图思路：一条主链条 + 若干小示意，结构化即可，文字可以先写，图稿我也能帮你画流程框架）

### 1）SumIR 生成流程

* **输入**：原生模块（C/C++ 编译后的 ELF / LLVM Bitcode），通过 Node-API 与 ArkTS 绑定。

* **步骤**：

* 1. **绑定解析**：解析 napi\_define\_properties、napi\_set\_named\_property 等绑定点，确定 ArkTS 名称 ↔ 原生函数地址的对应关系。

* 1. **函数切片**：定位到具体的原生函数实现，对其中的数据依赖进行静态分析。

* 1. **三类指令抽象**（SumIR）：

* + - **CALL**：表示一次函数调用或 Node-API 调用的输入输出依赖（包括参数传递、句柄解引用）。

* + - **PHI**：表示控制流合流点的数据依赖（保留路径敏感语义）。

* + - **RET**：表示函数返回的数据语义（支持多返回值和指针出参）。

* 1. **摘要生成**：将完整函数的数据流关系压缩为 SumIR 摘要。

* **输出**：SumIR 文件，独立于源码/二进制输入，统一结构，支持跨语言抽象。

（图示：左边原生函数，箭头到“绑定解析器”，再到“数据流分析”，生成右边的“SumIR（三指令序列）”）

### 2）IR 翻译与无缝接入流程

* **输入**：SumIR 摘要 + ArkTS 调用点（ArkIR）。

* **步骤**：

* 1. **调用点识别**：在 ArkIR 中找到对应的 ArkTS 调用语句（通过绑定表匹配）。

* 1. **类型推断**：根据 ArkTS 语义推断 napi\_value 对应的真实类型（如 string、buffer、object）。

* 1. **指令映射**：逐条将 SumIR 翻译为 ArkIR 中的 SSA 语句：

* + - CALL → ArkIR 调用节点；

* + - PHI → ArkIR phi 节点（保持数据依赖）；

* + - RET → ArkIR return 节点（与调用点连接）。

* 1. **函数体重建**：在 ArkIR 内部插入一个合成函数体，表示原生函数的逻辑。

* 1. **调用边连接**：将 ArkTS 调用点与重建函数体相连，加入 CFG/CG，使后续分析器认为这是一个普通 ArkTS 函数。

* **输出**：扩展后的 ArkIR 程序，可直接交由既有分析器（污点追踪、数据流分析）处理。

（图示：左边 ArkIR 调用点，右边 SumIR，经过“类型推断+翻译器”模块，生成一个 ArkIR 函数体，最后回连到调用点）

## 二、核心保护点（建议写法）

1. **统一抽象层**：通过定义 SumIR（CALL/PHI/RET）作为原生代码的数据流摘要，实现跨语言一致性表示。

1. **类型驱动翻译**：在 ArkTS 端通过类型推断机制，将 SumIR 自动映射为 ArkIR 函数体。

1. **无缝接入**：将重建的函数体与调用点连边，嵌入 ArkIR 的控制流图，使原生逻辑在既有分析器中透明可见，无需额外改造。

1. **双输入适配**：无论输入为二进制 ELF 还是源码 Bitcode，最终均生成统一的 SumIR → ArkIR 流程。

1. **多返回与句柄建模**：SumIR 支持多返回值与 napi\_value 指针出参的抽象建模，这是区别于传统单返回函数摘要的关键点。

## 三、取证手段与效果

* **可验证性**：SumIR 文件可独立保存，作为“原生函数语义摘要”，具备可重复取证的特征。

* **一致性**：不同输入形态（源码/二进制）都会导出相同的 SumIR，便于跨版本或跨编译环境的对比取证。

* **覆盖性**：在 ArkIR 中自动还原原生函数语义，保证跨语言数据流在下游分析（例如泄露检测、隐私合规检查）中完整出现。

* **差异点**：现有 Android JNI 静态分析只做到“调用摘要”级别，无法与高层 IR 无缝接入；而本技术首次实现了 HarmonyOS Node-API 跨语言场景的“抽象—翻译—接入”的闭环。

## 小结（给你一句总领性表述）

本技术的独特之处在于：**提出了一种跨语言数据流抽象与翻译机制，将鸿蒙 Node-API 原生调用的数据依赖压缩为三类 SumIR 指令，并在 ArkTS 侧自动翻译为 ArkIR 函数体，最终与调用点无缝衔接**。通过这一机制，原本“黑箱”的原生逻辑被透明嵌入到现有分析框架中，无需对分析器本身进行任何修改，即可覆盖跨语言数据流。

要不要我帮你把这部分直接扩展成“专利说明书”里的 **“技术方案”+“有益效果”** 两大段落的完整文字？这样你可以直接拿去合并到申请稿里。

# 未命名

明白了 ✅。  
你的意思是：**取证手段**这一块，不能像论文那样写成“实验可重复”或者“覆盖性”，而是要直接服务于“核心保护点”——即 SumIR 生成 + IR 翻译无缝接入。换句话说，要强调：**如何通过技术特征来界定、证明和保护这两个点**。

我帮你重写一版，结构是：

1. **核心保护点与取证手段对应关系**

1. **可用于专利说明书的表述**

## 1. 核心保护点与取证手段对应关系

### （A）SumIR 生成

* **保护点**：提出一种最小完备的三指令抽象（CALL / PHI / RET），可统一表示 Node-API 原生函数的数据流。

* **取证手段**：

* + 检查生成的 SumIR 文件中是否包含三类指令及其数据依赖关系；

* + 验证 CALL 指令是否覆盖了 Node-API 参数提取与外部调用依赖；

* + 验证 PHI 指令是否对不同控制流分支进行了数据合流建模；

* + 验证 RET 指令是否正确反映了函数的多返回值/指针出参；

* + 对同一原生函数在不同输入形态（源码/二进制）下生成的 SumIR 是否一致。

* **意义**：这些 SumIR 摘要文件本身可作为可保存、可比对的取证材料，用以证明该方法确实对原生逻辑进行了统一建模。

### （B）IR 翻译与无缝接入

* **保护点**：基于类型推断，将 SumIR 自动翻译为 ArkIR 函数体，并与 ArkTS 调用点建立调用边，使原生逻辑透明进入控制流。

* **取证手段**：

* + 在 ArkIR 中定位由本方法插入的合成函数体；

* + 检查该函数体内语句是否与 SumIR 指令一一对应（CALL→调用节点、PHI→phi 节点、RET→返回节点）；

* + 检查调用点与合成函数体之间是否存在新增的调用边；

* + 通过对比“翻译前后”的 ArkIR 控制流图，验证原本缺失的跨语言数据流是否已经被完整还原；

* + 在现有分析器（如污点追踪）运行时，验证其是否能识别跨语言的敏感数据路径。

* **意义**：重建函数体和调用边的存在是直接证据，证明该方法不仅生成摘要，而且**将摘要语义无缝注入分析生态**。

## 2. 可用于专利说明书的表述（示例）

**取证手段**  
本发明的核心技术特征可通过以下手段加以验证和界定：

1. **SumIR 摘要验证**：对目标原生函数执行本方法后，输出的摘要应包含由 CALL、PHI 和 RET 三类指令构成的数据流序列。通过检查该摘要可确认是否对参数提取、分支合流和返回值进行了完整建模。进一步地，针对同一函数的不同输入形态（源代码或二进制），其生成的 SumIR 应保持一致性。

1. **ArkIR 函数体验证**：在 ArkIR 程序中，应用本方法后应出现与 SumIR 一一对应的合成函数体。该函数体的指令与 SumIR 指令在语义上可逐项比对。

1. **调用边验证**：在 ArkIR 的控制流图中，原本仅表现为外部调用的节点，应新增与合成函数体之间的调用边。此调用边的存在可直接证明本方法实现了原生语义向 ArkTS 层的无缝接入。

1. **下游分析验证**：通过现有数据流分析工具在扩展后的 ArkIR 上运行，可检测到跨越 ArkTS 与原生模块的数据流路径。该结果与未应用本方法的情况进行对比，可直观证明无缝接入的效果。

**综上**，SumIR 文件、ArkIR 合成函数体、调用边以及分析结果均可作为可保存、可比对的客观证据，用以证明和保护本发明的技术特征。

要不要我帮你把这一部分，直接写进完整的“**发明内容 + 技术方案 + 有益效果 + 取证手段**”四段落的专利说明书版本？这样你就可以直接替换现有初稿了。