SCA-GPT: A Generation-Planning-Tool Assisted LLM Agent for Fully Automated Side-Channel Analysis on Cryptosystems

Wenquan Zhou, An Wang, Yaoling Ding, Annyu Liu, Jingqi Zhang, Jiakun Li, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—Non-invasive security testing under standards such as ISO/IEC 17825 still relies on human experts: how to measure a given target, and what the standard requires once the results are available, differ with every device and algorithm class, so the procedure is composed anew for each assessment. We present *SCA-GPT*, a large language model (LLM) agent for side-channel evaluation: given only a single natural-language instruction, it completes a full standard-conformant assessment covering timing analysis, simple power analysis, and differential power analysis, reporting for every test item a verdict, the governing clause, and checkable numeric evidence. The agent integrates a domain-specific expert knowledge base curated from 155 documents with a suite of specialized side-channel analysis tools. In retrieval experiments, the expert knowledge base achieves a Hit Rate@5 of 89.7% and an MRR@5 of 74.1%. We further evaluate the full framework with three LLMs on 13 real datasets covering six cryptographic algorithms on smart-card, microcontroller, and FPGA targets. DeepSeek-V3.2, Kimi-K2.6, and Qwen3.5 achieve success rates of 90.0%, 87.7%, and 94.6%, respectively, versus 0.0% for a plain ReAct agent, at an average of 253 seconds per assessment. To our knowledge, *SCA-GPT* is the first automated LLM agent for standard-conformant side-channel evaluation.**

***Index Terms*—Side-channel analysis, large language model, retrieval-augmented generation, expert knowledge base, automated standard-conformant security testing.**

I. Introduction

In recent years, large language models (LLMs) such as ChatGPT [1], LLaMA [2], and DeepSeek [3] have been successfully applied in numerous domains, including code generation [4], vulnerability management [5], and anomaly detection [6]. Emergent capabilities such as in-context learning [7] and chain-of-thought reasoning [8] allow them to solve complex tasks in zero-shot or few-shot settings, and frameworks like ReAct [9] extend these capabilities toward human-like planning and decision-making. Still, LLMs by themselves cannot access up-to-date information [10] or reliably perform precise numerical computation [11]. LLM agents overcome these limits through tool use: by invoking external tools, a model can retrieve information, execute complex operations, and act on systems in the physical world [12], [13].

Hardware and embedded-system security is critical. To ensure this security, products incorporating cryptographic modules must undergo rigorous security testing before entering the market. As a principal non-invasive attack technique, side-channel analysis (SCA) extracts keys and other sensitive information by monitoring physical leakages such as power consumption, electromagnetic emissions, or timing, and can defeat cryptographic implementations without attacking the underlying algorithm [14], [15], [16], [17]. Since the 1990s, researchers and industry have proposed various SCA techniques such as timing analysis (TA), simple power analysis (SPA), and differential power analysis (DPA) [18], [19]. Given this threat, international standards have made resistance to non-invasive attacks a required test item. For example, FIPS 140-3 introduced requirements for non-invasive physical attacks and explicitly states that these attacks obtain secret information from cryptographic modules via external measurements [20]; ISO/IEC 17825 further defines a black-box test vector leakage assessment (TVLA) for evaluating resistance to power and electromagnetic leakage [21], [22].

However, existing side-channel security evaluations still depend on experienced testers, customized software, and large amounts of measurement data. The literature notes that despite various attempts at automation, security testing of embedded systems remains predominantly manual [23]; in SCA such manual testing is especially time-consuming and costly, with expert resources scarce and expensive. Tool-based automation stops short of the judgment: surveys of the design-tool landscape [24] and systematic evaluations of automated detectors [25] document that such tools automate *detection* while the interpretation, and even the choice of which tool to run, remain expert tasks. This manual dependence drives inconsistent conclusions across analysts [26]. A program can encode that judgment, but only for the situations its author anticipated: every target differs in trace morphology, clock rate, and record length, and what to do next follows from what the previous item returned. Encoding every branch is writing expert knowledge into code, which is workable for a fixed set of targets but not for the open set a laboratory receives. The standard's own acceptance criteria compound the difficulty: the fixed ± 4.5 threshold of ISO/IEC 17825 admits both false negatives and false positives [27], the fixed-versus-random t-test can miss real leakage [28], and the test-vector construction does not transfer cleanly to newer primitives [29]. Deciding which criterion applies to which target therefore demands expert reasoning rather than a fixed recipe.

This work was supported in part by the National Natural Science Foundation of China under Grant 62272047 and Grant 62502035, in part by the Beijing Natural Science Foundation under Grant L244044 and Grant L251068, and in part by the State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, under Grant KFZD2503. (Corresponding author: Yaoling Ding.)

Wenquan Zhou, An Wang, Yaoling Ding, Annyu Liu, Jingqi Zhang, Jiakun Li, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: dyl19@bit.edu.cn). Wenquan Zhou and An Wang are also with the State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao 266237, China.

Wenquan Zhou and An Wang contributed equally to this work.

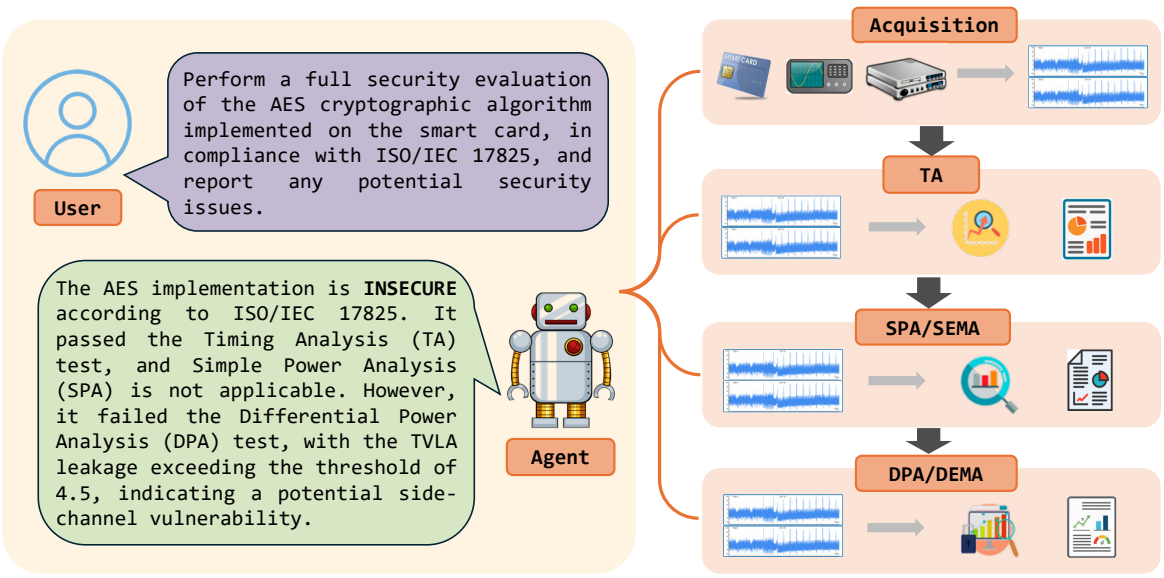


Fig. 1. LLM agents performing automated non-invasive security evaluation.

This reliance on manual expert procedures creates critical challenges throughout the evaluation process. First, expert variability introduces inconsistent evaluation outcomes, as different experts may interpret standards differently or employ varying analysis strategies, leading to results that lack reproducibility. Second, the complex, multi-step nature of ISO/IEC 17825-conformant testing makes manual execution inherently error-prone. Third, many device manufacturers lack in-house SCA expertise and must outsource evaluations to third-party laboratories, raising concerns about intellectual property leakage while incurring substantial testing costs. These handcrafted processes make it difficult to cover all devices and environments and impede large-scale compliance testing. Thus, although the community envisions a "push-button" style leakage test [23], mature automated SCA tools that deliver consistent, expert-level evaluations remain urgently needed.

Since LLM agents are capable of tool usage, could they also be applied, as illustrated in Fig. 1, to achieve precise and fully automated ISO/IEC 17825-conformant non-invasive security evaluation?

Applying LLMs to SCA has been an active line of work since 2023, yet none of it automates a standards-based evaluation. Most efforts steer the LLM toward *attack*: AgentSCA [30] and Waligóra [31] pursue key recovery over physical measurements. The remainder automates one isolated stage rather than an evaluation: Netlist Whisperer [32] predicts leaky cells before silicon, ThreatLens [33] drafts hardware-security test plans but does not execute them, and SCAgent [34] discovers software side channels. The picture repeats in security testing at large: end-to-end security agents such as PentestGPT [35] are built for exploitation, not for compliance. What a testing laboratory needs is different: a system that executes the complete procedure a standard prescribes, from raw traces through every required test item to an evaluation report, and issues a clause-grounded pass/fail verdict with no expert in the loop. Table I places these systems along four capabilities a compliance evaluation requires: operating on physical traces (Phys.), executing the analysis (Exec.), grounding decisions in retrieved knowledge (Knowl.), and delivering the compliance evaluation itself (Compl.). Each prior system covers some of them, but none closes the chain; *SCA-GPT* covers all four.

To fill this research gap, we present *SCA-GPT*, a knowledge-infused LLM agent framework designed for fully automated non-invasive security evaluation of hardware and embedded systems. However, realizing this vision requires overcoming two fundamental challenges that prevent direct application of existing LLM technologies to the SCA domain.

TABLE I
CAPABILITY COMPARISON OF LLM-ASSISTED SIDE-CHANNEL SYSTEMS.

Work	Phys.	Exec.	Knowl.	Compl.
AgentSCA [30]	✓	✓	✗	✗
Waligóra [31]	✓	✓	✗	✗
ThreatLens [33]	✗	✗	✓	✗
SCAgent [34]	✗	✓	✗	✗
SCA-GPT (ours)	✓	✓	✓	✓

First, current foundation models are designed for language-to-language transformations and cannot directly process side-channel traces. While general-purpose tool-augmented agents can invoke external programs, existing agent frameworks lack specialized SCA tools required for domain-specific tasks such as power trace preprocessing, leakage quantification, and statistical distinguisher implementation. Without these purpose-built instruments, even advanced LLMs cannot bridge the gap between abstract reasoning and concrete SCA operations.

Second, a standard-conformant evaluation demands expertise that current LLMs do not reliably possess: which test item applies to which algorithm class, in what order the tests gate one another, and which clause governs each verdict.

Placing standards documents in the context window does not by itself make them operative. A document supplies the same text whatever the situation, whereas different situations call for different knowledge: what must surface is the material that the decision at hand depends on, at the moment it is needed. The fragmented principles scattered across standards, literature, and expert practice must therefore be consolidated into structured, retrievable entries. Building such an expert knowledge base, and retrieving from it precisely throughout a multi-step evaluation, is the second challenge.

To address these challenges, we make the following contributions:

- **A standard-conformant evaluation agent.** We present *SCA-GPT*: given only a single natural-language instruction, it completes a full ISO/IEC 17825 assessment covering TA, SPA, and DPA, and reports, for every test item, a verdict together with the governing clause and the numeric evidence behind it. Three modules divide the work: a tools module supplies the low-level computation the standard prescribes, an expert knowledge base supplies what each decision depends on, and a planning module constructs the analysis for the target at hand. To our knowledge, it is the first automated LLM agent for standard-conformant side-channel evaluation, as opposed to attack, leakage detection, or test planning.
- **An expert knowledge base validated on real queries.** We construct a domain-specific expert knowledge base of 155 documents (11,449 chunks) together with hand-crafted expert entries, embedded in the agent's long-term memory. On queries drawn from the agent's own retrieval calls, the retriever reaches a Hit Rate@5 of 89.7% (against 54.3% for a random baseline) and a mean reciprocal rank (MRR@5) of 74.1%.
- **Extensive experimental validation.** We evaluate the framework on 13 real datasets covering six cryptographic algorithms on smart-card, microcontroller, and FPGA targets, with three LLMs and four configurations, including two ablations. *SCA-GPT* achieves success rates of 90.0% (DeepSeek-V3.2), 87.7% (Kimi-K2.6), and 94.6% (Qwen3.5) at an average of 253 seconds per assessment, whereas a plain ReAct agent with a Python interpreter, the plain combination of existing components, ends at 0.0%.

II. Background

A. Non-Invasive Security Evaluation Standard

In recent years, with the widespread deployment of cryptographic algorithms in smart cards, IoT devices, and embedded control systems, non-invasive security evaluation has become increasingly important. SCA enables the recovery of cryptographic keys by exploiting physical leakages without damaging the hardware, and has emerged as a core technique for assessing the resistance of cryptographic implementations against attacks. TA, simple power/electromagnetic analysis (SPA/SEMA), and differential power/electromagnetic analysis (DPA/DEMA) are among the most commonly used SCA methods and have been successfully demonstrated against both commercial and security-grade chips; recent work drives them with neural distinguishers [36].

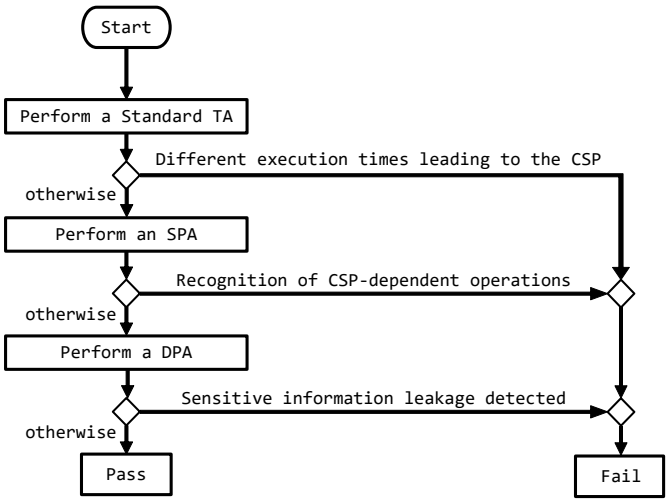


Fig. 2. Side-channel resistance test framework.

To raise the security and compliance requirements for cryptographic products, the International Organization for Standardization introduced ISO/IEC 17825 to unify evaluation methodologies and promote standardized testing procedures. This standard specifies a sequential testing workflow for assessing side-channel resistance (as illustrated in Fig. 2): each test asks whether a critical security parameter (CSP), the standard's term for keys and other secrets, is exposed through the corresponding channel, and test laboratories must follow the prescribed order. For example, SPA/SEMA can only be conducted if TA has passed. Each category of attack, namely TA, SPA/SEMA, and DPA/DEMA, is accompanied by detailed testing specifications. In particular, the DPA/DEMA section incorporates the TVLA methodology, whose black-box nature significantly simplifies the detection process and reduces implementation complexity.

The adoption of ISO/IEC 17825 has not only advanced the standardization and comparability of hardware security evaluations but has also provided a unified and actionable technical foundation for secure chip design, testing laboratories, and compliance certification. Despite these advances, conducting ISO/IEC 17825-conformant evaluations remains heavily dependent on domain experts, creating significant barriers for organizations seeking to assess their products. Existing attempts at automation either rely on semi-automatic locating of cryptographic operations in side-channel traces [37] or focus only on SPA in specific scenarios [38], still requiring substantial expert intervention throughout the evaluation process. This dependence on specialized expertise not only increases costs and time but also limits accessibility for device manufacturers who need rapid, accurate self-assessment capabilities during development cycles.

B. LLM-Powered Autonomous Agents

LLMs have advanced rapidly in natural language processing and beyond, demonstrating strong generalization capabilities

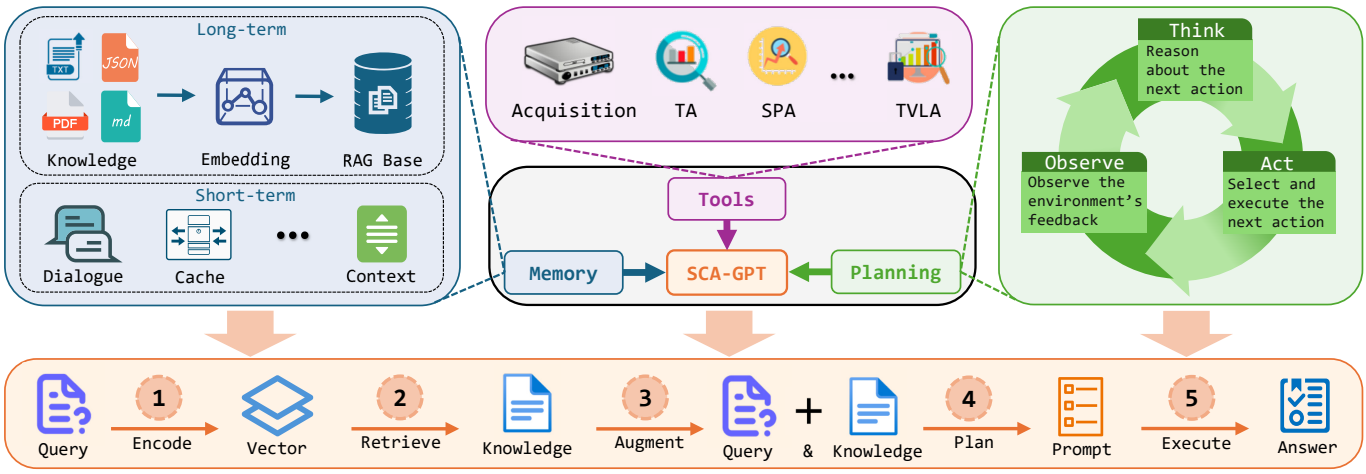


Fig. 3. The overall architecture of *SCA-GPT*. The ordinal numbers in the figure represent the order of execution.

across tasks and domains. However, early LLMs primarily operated in static text-interaction settings, functioning essentially as "passive response systems."

To overcome these limitations, the concept of the agent was introduced. Around a central LLM, an agent typically extends its reach in three ways: it invokes external tools to act on systems beyond its parameters, it retains and recalls information beyond the context window through external storage and retrieval, and it organizes its reasoning into iterative decision cycles. Techniques such as chain-of-thought prompting [8] and Tree-of-Thoughts exploration [39] structure the reasoning itself, and ReAct [9] closes the loop between reasoning and acting, letting the model interleave natural-language thought with concrete tool calls.

Recent proof-of-concept systems, such as AutoGPT [40] and GPT-Engineer [41], have demonstrated the potential of LLM-based agents. Exploratory applications in other domains, including ChemCrow [42] for chemical research and Generative Agents [43] for simulating complex interactive behaviors, further validate the feasibility of the agent paradigm. These examples show that agent architectures suit domains requiring high decision accuracy and execution capability, such as security testing, scientific computing, and automated operations. In the field of hardware security in particular, the autonomy and scalability of LLM-based agents lay a solid foundation for building fully automated, end-to-end security evaluation frameworks. Motivated by these advances, we aim to leverage LLMs to address the dual challenges of tool unavailability and expert knowledge scarcity in SCA, ultimately delivering an autonomous framework capable of ISO/IEC 17825-conformant evaluations without expert intervention.

III. A Generation-Planning-Tool Assisted LLM Agent

A. Overview and Motivation

SCA-GPT is designed to complete a full ISO/IEC 17825 evaluation given only a single natural-language instruction. Three obstacles stand between a plain LLM agent and this goal. **(C1) Computation must be reliable.** The statistics the standard prescribes run over tens of thousands of traces; analysis code improvised by the model at this scale is error-prone, and a single wrong number invalidates the verdict built on it. **(C2) Expert knowledge must be retrievable on demand.** Which test applies to which algorithm class, in what order the tests gate one another, and which clause governs each verdict are specialist facts that general-purpose LLMs absorb only partially during pretraining, since material of this kind is sparse in their training corpora; the agent must be able to look them up in a curated source at the moment a decision depends on them. Because each target and each algorithm class raises different questions, what is needed is not a body of text held in context but different knowledge drawn for different situations. **(C3) The outcome must be auditable.** A compliance verdict is worthless without the clause it rests on and the numeric evidence behind it, so the agent must end every run in a structured, checkable report rather than free text. These three obstacles map onto the three modules of *SCA-GPT*: the tools module answers C1 with deterministic specialized operators, the expert knowledge base held in the memory module answers C2, and the planning module answers C3 by driving the run autonomously, iterating until the assessment can close with a structured submission. Fig. 3 shows the resulting architecture; the three modules (memory, tools, and planning) are detailed in Section III-B, and Section III-C then presents the complete workflow of an evaluation run.

B. Agent Modules

1) Memory: The memory module comprises both short-term and long-term memory. Short-term memory holds transient information such as the current task context, dialogue history, intermediate tool outputs, and local caches, but it is inherently limited by context-window size, volatility, and lack of curated domain structure: limitations that prevent it from reliably improving task success on its own. To address these shortcomings, *SCA-GPT* embeds a curated, domain-specific expert knowledge base into the long-term memory via a retrieval-augmented generation (RAG) mechanism. This knowledge base, constructed from standards, handcrafted

expert entries, and public literature, is model-agnostic and provides persistent, provenance-rich context that the LLM can query both at the outset of the task and dynamically during intermediate reasoning steps, thereby materially improving planning, verification, and standard-conformant decision making. As the centerpiece of the framework, this expert knowledge base is usable with any LLM; in the side-channel domain, its construction methodology and quality are critical determinants of the agent's overall capability.

An agent's capability rests not only on reasoning and planning but on its command of high-quality domain knowledge, and the expert knowledge base is what carries it. The knowledge base is also inherently extensible: it can be continuously enriched with new standards, experimental experience, and expert entries, progressively broadening its coverage of side-channel security evaluation.

We define a unified representation for all knowledge entries:

$$\mathbf{v}_i = f_\theta(E_i), \quad E_i \in \mathcal{D}, \tag{1}$$

where $\mathcal{D}$ denotes the collection of knowledge sources, including both *document-based knowledge* and *handcrafted expert entries*. Here, E_i is the i-th knowledge entry extracted from $\mathcal{D}$. The function $f_\theta(\cdot)$, typically implemented by a pre-trained embedding model such as bge-base-en-v1.5, maps E_i into a high-dimensional semantic space. The resulting semantic vector is $\mathbf{v}_i \in \mathbb{R}^d$, where the dimensionality d (e.g., 768 for bge-base-en-v1.5) is determined by the specific embedding model configuration.

Document-based knowledge includes international standards (e.g., ISO/IEC 17825), research papers, textbooks, experimental manuals, and technical blogs. These materials are segmented into chunks and transformed into vector embeddings by f_θ, which are then stored in the vector database.

Handcrafted expert entries are written by researchers to emphasize logic and structure. Formally, each entry is represented as:

$$E_i = \{t, [s_1, s_2, \ldots, s_n]\}, \tag{2}$$

where t is the thematic label and $[s_1, s_2, \ldots, s_n]$ is an ordered sequence of sub-steps. This structured formulation ensures consistency and extensibility of expert entries.

2) Tools: Beyond accurate reasoning, effective evaluation also depends on precise execution. While conventional agents may generate and run code, this approach alone often leads to high error rates or repeated failures on complex programming subtasks. To mitigate this, *SCA-GPT* integrates a suite of specialized SCA tools, as summarized in Table II. These include the trace-acquisition, TA, and TVLA tools. In addition, a Python interpreter tool is provided, enabling the LLM to implement custom analytical workflows when dedicated tools are unavailable. SPA is a case in point: recovering an operation sequence from a single trace depends on the trace morphology of the target at hand, so the agent composes this analysis in the interpreter rather than calling a fixed operator. The toolset spans the full pipeline from trace acquisition to leakage analysis; guided by both the embedded expert knowledge and these tools, *SCA-GPT* produces conclusive security assessments in a fully automated manner.

TABLE II
SPECIALIZED SIDE-CHANNEL ANALYSIS TOOLS.

Tool	Description
TVLA	Implements test vector leakage assessment for statistical detection of side-channel leakage in a black-box setting.
Retriever	Retrieves the most relevant knowledge from the expert knowledge base using approximate nearest-neighbor search and summarizes the content.
TA	Measures and analyzes execution time variations to detect timing-based leakage.
Trace acquisition	Captures high-resolution power or electromagnetic traces from target devices for leakage analysis.
Python interpreter	Executes Python scripts to perform custom analysis or automation tasks when no dedicated tool is available.

3) Planning: The planning module serves as the central decision-making engine of *SCA-GPT*, orchestrating task execution through a structured *think–act–observe* loop. Upon receiving the user's query and the expert knowledge retrieved from the expert knowledge base, the planning module prompts the LLM to engage in iterative cycles of reasoning and action. In each cycle, the LLM first analyzes the current goal or problem context (*think*), determines the most suitable next action (*act*), and subsequently observes the feedback or output from the environment (*observe*). Actions may include invoking a specialized tool, generating and executing code, or querying additional knowledge. Because the retrieval tool is itself one of the available actions, the standard can be re-queried at any intermediate decision rather than only at the outset of the task. This process is repeated, with each observation informing the next round of reasoning, until the task objectives are achieved and the final answer is produced.

C. End-to-End Workflow

An evaluation run begins with the task instruction and ends with a submitted report; Fig. 3 traces its five steps. The instruction is first encoded into a query vector (①), the expert knowledge base is searched for the governing procedure and clauses (②), and the retrieved knowledge is fused with the running context (③). The run then enters the think–act–observe loop: each cycle builds the working prompt (④), chooses and executes one action (⑤)—a specialized tool, generated code, or a fresh knowledge query—and feeds the observation back into the next decision through short-term memory.

Algorithm 1 states the procedure this loop follows. The retrieved knowledge fixes the ordered test flow, and the run traverses it in the standard's prescribed order: each applicable item is settled by a think–act–observe loop and recorded with its governing clause and evidence, while inapplicable items are marked directly from the standard. The traversal is gated: a failing item that the standard treats as a gate already establishes non-compliance and stops the run, which is how ISO/IEC 17825 sequences TA before SPA before DPA.

To make this concrete, the following walks through one pass of Algorithm 1: a full ISO/IEC 17825 evaluation of an AES implementation on a contact smart card.

Algorithm 1 *SCA-GPT* standard-conformant evaluation

Input: a single natural-language instruction q; fixed system resources: expert knowledge base KB, tool library $\mathcal{T}$
Output: structured report $R = \{(\tau, \text{verdict}, \text{clause}, \text{evidence})\}$, one tuple per test item

$K \leftarrow \text{RETRIEVE}(\text{KB}, \text{ENCODE}(q))$ ▷ test flow and governing clauses
$F \leftarrow$ ordered test items in K ▷ e.g., ⟨TA, SPA, DPA⟩
$R \leftarrow \emptyset$; $H \leftarrow \emptyset$ ▷ report; short-term history
for each item $\tau \in F$ in the prescribed order **do**
  if τ is not applicable to the algorithm class **then**
    $R \leftarrow R \cup \{(\tau, \text{N/A}, \text{clause}(\tau, K), -)\}$
    continue
  end if
  repeat
    $a \leftarrow \text{PLAN}(q, K, H, \mathcal{T})$ ▷ choose an action from $\mathcal{T}$: a tool, code, or a fresh KB query
    $o \leftarrow \text{EXECUTE}(a)$
    $H \leftarrow H \cup \{(a, o)\}$
  until τ is concluded **or** the iteration budget is exhausted
  if τ is not concluded **then**
    $R \leftarrow R \cup \{(\tau', \text{INCONCLUSIVE}, \text{clause}(\tau', K), -) : \tau' \text{ from } \tau \text{ onward}\}$; **break** ▷ the run is unsuccessful
  end if
  $(v, e) \leftarrow$ verdict and evidence extracted from H
  $R \leftarrow R \cup \{(\tau, v, \text{clause}(\tau, K), e)\}$
  if $v = \text{FAIL}$ **and** τ gates the remaining items **then**
    $R \leftarrow R \cup \{(\tau', \text{N/A}, \text{clause}(\tau', K), \text{gated}) : \tau' \text{ after } \tau \text{ in } F\}$
    break ▷ a failed gate already establishes non-compliance
  end if
end for
submit R ▷ the structured report closes the run

- **Query**: perform a full ISO/IEC 17825 security evaluation of the AES implementation on the smart card, and report any security issues.
- **Prompt**: the query is augmented with the retrieved test-flow entry, the governing clauses, and the tool descriptions (TA, TVLA, Python interpreter), forming the prompt that guides the agent's reasoning.
- **Cycle 1 (TA)**: *Think*: the standard requires TA first, and the later tests proceed only if it passes. *Act*: invoke the TA tool to compare the operation-time distributions of fixed and random inputs with Welch's t-test [44]. *Observe*: no significant timing difference; TA passes under clause 8.2.
- **Cycle 2 (SPA)**: *Think*: for a symmetric cipher, single-trace SPA cannot recover the key, which the retrieved clause confirms. *Act*: check SPA applicability against that clause. *Observe*: SPA is not applicable (clause 8.3.1); proceed to DPA.
- **Cycle 3 (DPA)**: *Think*: the next test is TVLA over the fixed-versus-random trace set at the ± 4.5 threshold. *Act*: invoke the TVLA tool on the 40,000-trace set. *Observe*: the peak t-statistic reaches 315.2, far above 4.5; leakage is detected and DPA fails.
- **Cycle 4 (Report)**: *Think*: the flow is complete, so the verdict can be assembled together with its supporting evidence.
- **Final Answer**: the AES implementation is INSECURE under ISO/IEC 17825: it passes TA (clause 8.2), SPA is not applicable (clause 8.3.1), and it fails DPA (TVLA peak 315.2); the verdict is reported with the t-value trace and the underlying statistics.

From the instruction to the submitted verdict, the run involves no expert intervention: the procedure and its clauses come from retrieval, each measurement from a deterministic operator, and the gating and the final report from the loop itself.

IV. Experiments and Evaluation

A. Experimental Setup

The *SCA-GPT* agent framework runs on a workstation running Windows 11 Professional and equipped with a 13th-Gen Intel Core i5-13600K CPU at 3.50 GHz, 32 GB RAM, and an NVIDIA RTX 3080 GPU with 12 GB VRAM, providing sufficient computational capacity for the agent framework, embedding-based retrieval, and side-channel trace processing.

Three LLMs are employed: DeepSeek-V3.2, Kimi-K2.6, and Qwen3.5-397B-A17B (hereafter Qwen3.5). All models are accessed via the SiliconFlow API and queried with temperature 1.0 and a 16,384-token output limit. Temperature is left at 1.0 rather than lowered toward deterministic decoding, so that the framework is measured under the sampling conditions of ordinary use.

All traces are acquired with a shared setup: a PicoScope 3403D oscilloscope and MMSCA-101 acquisition modules capture the power and electromagnetic signals, while the coupling front end varies with the target—an ACR1281U reader for contact and contactless smart cards and direct probing

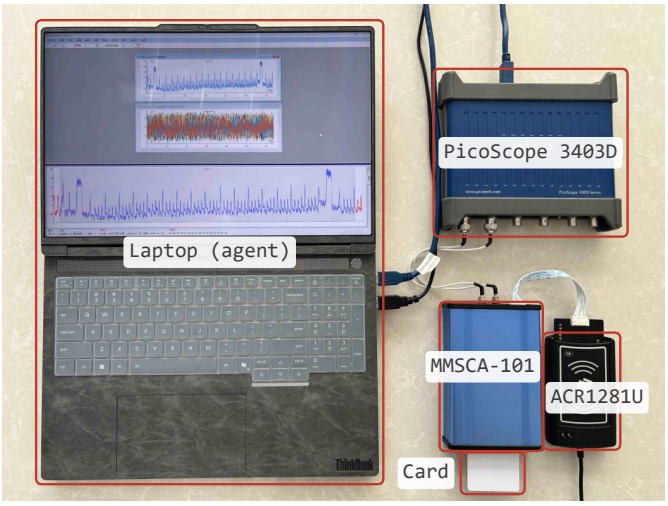


Fig. 4. Measurement setup for a contact smart-card target.

for the microcontroller and FPGA boards. Fig. 4 shows the topology for a contact smart-card target as a representative example, with the card under test inserted into the MMSCA-101 module. The framework can drive this setup directly through its trace-acquisition tool; the assessments reported here start from the captures submitted with the sample, which is the form in which measurement data reaches a laboratory.

We evaluate *SCA-GPT* on 13 datasets spanning six cryptographic algorithms (AES, DES, SM4, HMAC-SM3, RSA, ECDSA), implemented on contact and contactless smart cards, STM32F103/F429 and AT89S52 microcontrollers, and the Sakura-G FPGA; all traces are real acquisitions from these devices, and the four asymmetric sets were newly captured on the platforms of our earlier SPA study [38]. The set is deliberately diverse: clock rates span 4.8 to 25 MHz and record lengths span three orders of magnitude, from 1,500 to 451,129 points per trace; each of these targets would traditionally have had its analysis written for it by hand. The leakage label of each dataset is established independently of the agent: DPA labels are computed by a reference implementation of the ISO/IEC 17825 TVLA procedure on the same captures, and for SPA the ground-truth operation sequence is derived deterministically from the vendor-supplied key according to the square-and-multiply (RSA) or double-and-add (ECDSA) structure of the implementation, rather than by manual inspection of traces. Under this ground truth, the symmetric and hash datasets exhibit DPA leakage and the asymmetric datasets exhibit SPA leakage. This reference procedure encodes the same ISO/IEC 17825 workflow a human evaluator carries out by hand, so agreement with it is agreement with the expert assessment it codifies; the labels are reference-derived rather than the product of a separate subjective annotation. The ground truth for a dataset is therefore not a single statistic but the whole chain the standard prescribes (the TA verdict and its clause, whether SPA applies and what it establishes, and whether DPA detects leakage and at what peak); a run is correct only when it matches that chain. Implementations that leak are the discriminating case: on an implementation with no leakage almost any procedure arrives at a pass, whereas establishing leakage requires the prescribed measurement to be carried out and reported with the evidence behind it. Both sides of the verdict are nonetheless exercised: exactly one scored item in each run is a fail, and a run counts as correct only when the remaining items are resolved to pass or not applicable under the governing clause. Table III summarizes the datasets; capture sizes are given as traces × points. Each asymmetric capture is a single execution used for SPA; these datasets carry no separate timing capture, and their TA item is scored on the verdict and its governing clause alone.

Each run is driven by a single natural-language task instruction carrying only what a vendor submits together with a sample: the capture files and their acquisition parameters, plus, for asymmetric implementations, the parameters of the vendor-supplied key, such as its bit length and the operation counts it determines. It contains no test procedure, no statistical threshold, and no clause of the standard; supplying these key parameters is precisely what distinguishes a white-box compliance evaluation, where the key accompanies the sample, from an attack. The captures are acquired beforehand on the setup above, with fixed and random inputs interleaved in random order during acquisition and regrouped by class in the stored files; the evaluated loop starts from these files. The instruction for the AES–Contact smart card dataset reads as follows.

Query (AES–Contact smart card)

```
Evaluate whether the implementation described
below conforms to ISO/IEC 17825.

The implementation under evaluation is AES-128
running on a contact smart card (ISO 7816), with
a device clock of 4800000 Hz. Two captures are
provided under trace_data/.

Timing analysis capture: TA/AES_7816_TA_fixed&
random_5M_2000traces_198955points_cut.trs --
sampled at 5000000 Hz, 2000 traces of 198955
points each. The first 1000 traces were captured
with a fixed input, the remaining 1000 with
random inputs.

Leakage-detection capture: TVLA/AES_7816_TVLA_
fixed&random_5M_40000traces_4187points.trs --
sampled at 5000000 Hz, 40000 traces of 4187
points each. The first 20000 traces were
captured with a fixed input, the remaining 20000
with random inputs.

Dataset name for trace_io, if you use it:
AES_7816
```

B. Construction and Validation of the Expert Knowledge Base

The knowledge base supplies the domain knowledge that the agent retrieves at run time. It contains only generic material (standard clauses, test procedures, and public literature) and no conclusion about any implementation under test: the agent must map the generic rules onto the concrete traces by itself. It is a source of knowledge, not a source of answers.

The corpus comprises 155 documents (8,414,834 characters) drawn from seven categories of sources, as summarized in Table IV. In addition, four expert entries, one for the overall ISO/IEC 17825 test flow and one for each of TA,

TABLE III
DATASETS USED IN THE EXPERIMENTS.

Class	Dataset	Leakage	Clock (MHz)	TA	SPA	DPA
Symmetric	AES–Contact smart card	DPA	4.8	$2{,}000 \times 198{,}955$	–	$40{,}000 \times 4{,}187$
	AES–Contactless smart card	DPA	13.56	$2{,}000 \times 73{,}918$	–	$40{,}000 \times 5{,}000$
	AES–STM32F429	DPA	25	$2{,}000 \times 49{,}602$	–	$40{,}000 \times 7{,}500$
	AES–STM32F103	DPA	8	$2{,}000 \times 36{,}001$	–	$40{,}000 \times 15{,}000$
	AES–Sakura-G (FPGA)	DPA	24	$2{,}000 \times 1{,}500$	–	$40{,}000 \times 2{,}500$
	DES–Contact smart card	DPA	4.8	$2{,}000 \times 62{,}379$	–	$40{,}000 \times 13{,}000$
	DES–Contactless smart card	DPA	13.56	$2{,}000 \times 11{,}627$	–	$40{,}000 \times 4{,}000$
	SM4–Contact smart card	DPA	4.8	$2{,}000 \times 25{,}438$	–	$40{,}000 \times 5{,}000$
Asymmetric	RSA–Contact smart card	SPA	4.8	–	$1 \times 451{,}129$	–
	RSA–STM32F429	SPA	25	–	$1 \times 418{,}782$	–
	ECDSA–Contact smart card	SPA	4.8	–	$1 \times 106{,}752$	–
	ECDSA–AT89S52	SPA	12	–	$1 \times 49{,}248$	–
Hash	HMAC-SM3–Contact smart card	DPA	4.8	$2{,}000 \times 62{,}036$	–	$40{,}000 \times 8{,}000$

SPA, and DPA, are written manually in the structured format of (2), so that the agent retrieves not only isolated facts but also the procedural logic that connects them. All material is segmented into chunks of 1,000 characters with an overlap of 100 and embedded with bge-base-en-v1.5 (768 dimensions, cosine distance), yielding 11,449 chunks in the index. The expert entry for the overall test flow reads as follows.

Expert entry (ISO/IEC 17825 | Overall Test Flow)

```
Step 1: Timing Analysis (TA)
Determine whether timing analysis is applicable
to the implementation under test, then run the
measurement. Proceed to the next test only after
TA is resolved. The three tests are ordered and
gated: each one is carried out only if the
preceding one was passed. A failed test ends the
evaluation -- the implementation is
non-conforming from that point on, and the tests
that would have followed are not required.
Report each of those as not applicable and name
the test that failed as the reason, so the
report still accounts for every item in the
flow.
Step 2: Simple Power/EM Analysis (SPA/SEMA)
Determine applicability from the algorithm
class, then, where applicable, recover the
operation sequence from a single trace. Proceed
to the next test only after SPA is resolved.
Step 3: Differential Power/EM Analysis
(DPA/DEMA)
Apply a leakage-detection test over the full
trace set. This is the final test in the flow.
If SPA already established that the operation
sequence is distinguishable, the implementation
has failed and DPA is not required; report it as
not applicable rather than leaving it
unaddressed.
Step 4: Report
For every test item report: the verdict
(pass/fail), the clause of ISO/IEC 17825 the
verdict rests on, the numeric evidence produced
(statistics, sample counts, parameters), and the
reasoning. A verdict without its clause and its
supporting numbers is incomplete. Submit the
structured conclusion through the submit_result
tool; a run without it does not count as
finished.
```

TABLE IV
COMPOSITION OF THE EXPERT KNOWLEDGE BASE.

Source category	Documents	Characters
Academic papers	67	3,880,964
Open-source tools and tutorials	62	514,344
Standards (ISO/IEC, GM/T)	7	742,276
Evaluation guidance (SOG-IS, NIST)	6	781,044
Vendor tool manuals	5	1,844,516
Hardware manuals	4	77,687
Monographs, theses, lecture notes	4	574,003
Total	**155**	**8,414,834**

TABLE V
RETRIEVAL QUALITY OF THE EXPERT KNOWLEDGE BASE.

Retriever	Hit Rate@5 (%)	MRR@5 (%)
Random baseline	54.3	27.5
bge-base-en-v1.5	89.7	74.1
all-mpnet-base-v2	86.2	72.0
gte-base	**100.0**	**79.7**

We then validate the retrieval quality of the constructed knowledge base. The test questions are the questions the agent itself asks: from 980 recorded calls to the retrieval tool (`kb_search`), we cluster 247 distinct questions and sample 30 of them; one, whose answer lies outside the knowledge base, has no relevant chunk in the candidate pool and is excluded from scoring, leaving 29 effective questions. For each question, the reference answer, the set of chunks that actually answer it, is established by two annotators labeling the candidate chunks independently (raw agreement 82.0%, Cohen's $\kappa = 0.493$). Table V reports the results for three embedding models over the same annotated pool. With each query returning its top five chunks ($k = 5$), bge-base-en-v1.5 attains a Hit Rate@5 of 89.7% and an MRR@5 of 74.1%, and among the hits the first relevant chunk is ranked first in 73.1% of cases; a random reordering of the same candidate pool reaches only 54.3% and 27.5%. The bge-base-en-v1.5 model was fixed when the knowledge base was built and used for every run reported below; the other two were indexed afterward over the same corpus. All three clear the random

TABLE VI
MAIN RESULTS OVER 1,560 RUNS; ALL VALUES ARE PERCENTAGES.

	DeepSeek-V3.2			Kimi-K2.6			Qwen3.5		
Configuration	**Success**	**Verdict**	**Partial**	**Success**	**Verdict**	**Partial**	**Success**	**Verdict**	**Partial**
ReAct [9]	0.0	46.9	30.0	0.0	51.5	39.7	0.0	43.8	30.3
+Tools	0.8	47.7	38.5	0.0	50.0	42.6	0.0	48.5	39.1
+KB	54.6	72.3	80.3	80.0	98.5	92.8	50.8	77.7	80.0
SCA-GPT	**90.0**	**97.7**	**95.9**	**87.7**	**100.0**	**95.9**	**94.6**	**100.0**	**98.1**

baseline by a wide margin, and the retriever carrying the main experiments is not the strongest of them, so the results that follow do not rest on a favorable embedding choice.

C. SCA-GPT for Non-Invasive Security Evaluation

We now evaluate the complete framework. Each run receives one task instruction of the form shown in Section IV-A and must deliver the full ISO/IEC 17825 assessment following the workflow of Section II-A: TA compares the operation-time distributions of fixed and random inputs, SPA recovers the operation sequence from a single trace, and DPA applies the TVLA leakage test. For every test item the agent reports its verdict, the governing clause, and the supporting numbers, or its justified non-applicability. The full grid is 13 datasets $\times$ 4 configurations $\times$ 3 models $\times$ 10 repetitions $=$ 1,560 runs; the source code of the framework is publicly available.[1]

We compare four configurations. *ReAct* [9] is the plain-agent baseline: a ReAct loop with a general-purpose Python interpreter and no domain-specific operator, retrieval, or expert knowledge. *SCA-GPT* is the complete framework. Between them sit its two ablations: *+Tools* keeps only the specialized tool library, and *+KB* keeps only the retrieval over the expert knowledge base of Section IV-B. The two components are orthogonal: the tool library contains no path to the knowledge base, and the retrieval tool adds no domain operator. All four configurations share the same ReAct loop, Python interpreter, prompt template, and iteration budget (at most 40 iterations per run); they differ only in which components are enabled.

Every run is scored automatically against a ground truth that the agent cannot access, per test item and along two dimensions. The *verdict* dimension checks the reported pass/fail/not-applicable decision. The *process* dimension checks that the decision is properly grounded: the cited ISO/IEC 17825 clause must be the governing one, the claimed statistic must actually appear in the execution log, the measurement must follow the prescribed protocol, and the recovered SPA sequence must match the ground truth bit for bit. We report three aggregates. The *success rate* is the fraction of runs in which every scored item is correct in both dimensions (success means fully correct, not merely completed); *Verdict* is the same fraction with the process dimension ignored; and *Partial* is the mean fraction of scored items per run that are correct in both dimensions. An item whose ground truth is not applicable (SPA for a symmetric cipher) is still scored: the agent must declare it inapplicable and cite the governing clause; DPA for the asymmetric implementations, removed class-wide by the gate, is excluded from the denominators.

Table VI reports the main results, each cell aggregating 130 runs. The headline metric is the success rate because *Verdict* can be gamed: ground-truth verdicts are constant within an algorithm class, so a rule that maps symmetric implementations to DPA leakage and asymmetric ones to SPA leakage scores perfectly on *Verdict* without reading a single trace. An evaluation body accepts no verdict without its measurement, and the same rule scores 0.0% on the success rate; it is the success rate that separates assessment from label guessing, and all conclusions below are drawn from it.

The first and last rows of Table VI compare the plain agent with the complete framework. ReAct, the plain combination of an LLM and a Python interpreter, never delivers one fully correct assessment; *SCA-GPT* succeeds in 90.8% of runs averaged over the three models. The per-model columns show the gain is consistent across the three models tested: they lie within seven percentage points of one another under *SCA-GPT* (87.7%–94.6%), whereas under +KB they spread from 50.8% to 80.0%.

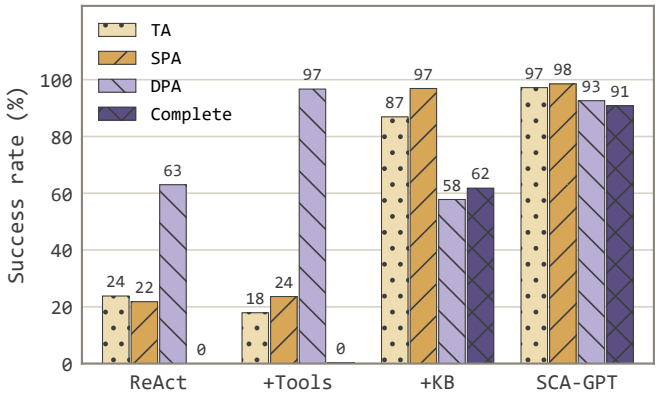
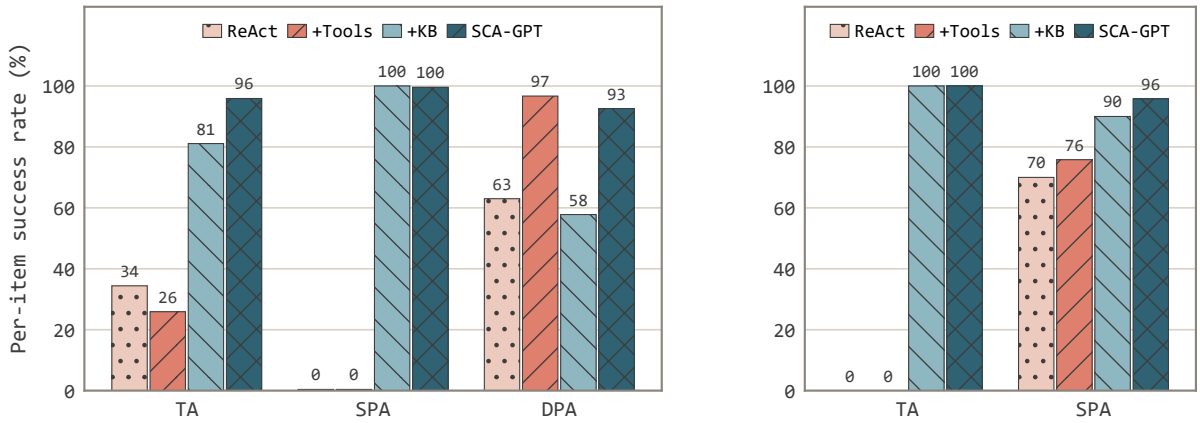


Fig. 5. Per-item and complete-assessment success rate by configuration, pooled over the three models.

Figs. 5 and 6 show what the tools and the knowledge base each contribute. Fig. 5 shows that ReAct's failure is not solely a computation problem: it is already fully correct on 63.0% of the DPA items, yet its *Partial* scores of 30.0%–39.7% mean roughly one correct item out of three; the per-item bars rise long before the complete-assessment bar leaves zero, and items and their assembly rise together only when knowledge and tools are both present.

[1] https://github.com/wenquan222/sca-gpt



(a) Symmetric and hash

(b) Asymmetric

Fig. 6. Per-item success rate by algorithm class.

Fig. 6 locates the two contributions by splitting the per-item success rate by algorithm class, counting the hash dataset with the symmetric ciphers since it takes the same DPA path. On asymmetric timing, ReAct and +Tools score 0.0% while +KB scores 100.0%: without the standard, the agent justifies its pass verdict with clause 8.2 of ISO/IEC 17825, which governs symmetric ciphers, instead of clause 9.3.2, which governs public-key timing; the number is right, the citation is wrong, and an assessment that cites the wrong clause is not an assessment. On symmetric SPA, +Tools scores 0.4% while +KB scores 100.0%: recognizing that single-trace SPA does not apply to a symmetric implementation is a matter of knowledge of the standard, not of computation. On symmetric DPA the roles reverse: +Tools reaches 96.7% while +KB drops to 57.8%, because a TVLA hand-coded on the fly is fragile whereas the deterministic operator is not. Computing correctly and speaking the standard are different capabilities; only *SCA-GPT* has both.

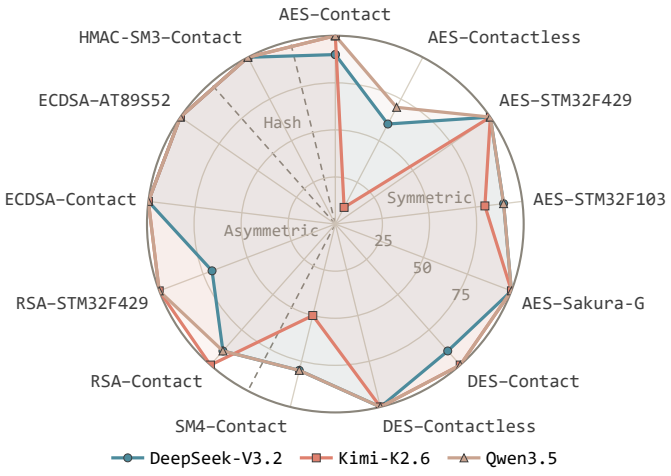


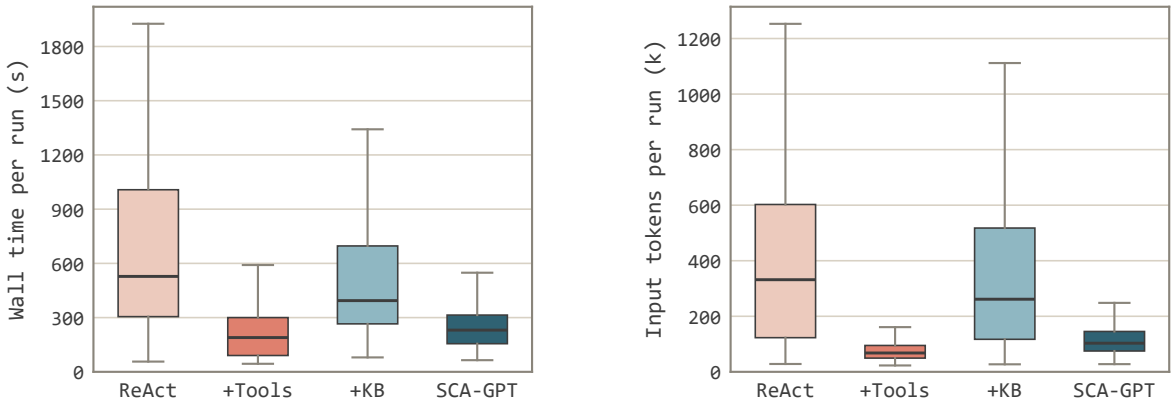
Fig. 7. Per-dataset success rate of *SCA-GPT*.

Fig. 7 breaks the results down to individual datasets: one spoke per dataset and one polygon per model for the complete framework, with dashed radii separating the symmetric, asymmetric, and hash classes; ReAct, which stays at zero on every dataset, would collapse to the center point and is omitted. *SCA-GPT* stays at or near the outer ring on nearly all datasets for all three models, and the residual failures are localized rather than systemic: Kimi-K2.6 scores a perfect 100.0% on 10 of the 13 datasets but collapses on AES–Contactless (10.0%), the single deep notch in the figure, which is what its standard deviation across datasets in Table VII records. The residual failures of *SCA-GPT* are process slips rather than wrong verdicts: the 36 failed runs contain 37 failed test items, one run failing two, and in 34 of the 37 the reported verdict is correct while the supporting process falls short; 16 of the 36 runs fall on AES–Contactless alone.

TABLE VII
RESULTS AND COST BY CONFIGURATION AND MODEL.

Configuration	Success rate (%)	SD	Wall (s)	Tokens in/out (k)	API calls
ReAct [9]	0.0	0.0	1,063	401.7 / 21.6	24.2
DeepSeek-V3.2	0.0	0.0	1,205	284.9 / 17.2	23.3
Kimi-K2.6	0.0	0.0	728	411.2 / 24.5	23.2
Qwen3.5	0.0	0.0	1,257	509.0 / 23.1	26.3
+Tools	0.3	0.9	**239**	**82.6** / 7.5	**10.1**
DeepSeek-V3.2	0.8	2.8	397	93.1 / 6.7	12.3
Kimi-K2.6	0.0	0.0	222	77.8 / 9.3	8.8
Qwen3.5	0.0	0.0	99	76.9 / 6.7	9.1
+KB	61.8	23.8	764	354.9 / 16.2	19.7
DeepSeek-V3.2	54.6	29.6	757	322.5 / 12.3	20.6
Kimi-K2.6	80.0	28.6	546	276.6 / 17.1	16.1
Qwen3.5	50.8	29.0	988	465.6 / 19.3	22.4
SCA-GPT	**90.8**	15.8	253	117.1 / **7.3**	10.5
DeepSeek-V3.2	90.0	12.9	279	130.5 / 5.3	12.1
Kimi-K2.6	87.7	27.4	287	124.1 / 9.8	10.5
Qwen3.5	94.6	9.7	193	96.7 / 6.7	8.8

Table VII and Fig. 8 report the cost; SD is the sample standard deviation of the per-dataset success rates, and each box in Fig. 8 pools the 390 runs of one configuration. A complete *SCA-GPT* run takes 253 s on average (median 231 s), with Qwen3.5 combining the best rate, the lowest variance, and the lowest token and time cost. Failure is not cheaper:



(a) Wall time

(b) Input tokens

Fig. 8. Per-run wall time and input tokens by configuration.

ReAct spends four times the time and over three times the tokens of *SCA-GPT* and never returns a complete assessment, its budget consumed by repeated failed attempts; +Tools finishes cheaply only because it answers without the retrieval and grounding work an assessment requires.

V. Conclusion

In this paper, we presented *SCA-GPT*, an LLM agent that completes ISO/IEC 17825 non-invasive security evaluations from a single natural-language instruction, reporting a clause-grounded verdict with checkable evidence for every test item. In an evaluation of 1,560 runs on 13 real datasets, *SCA-GPT* attains a 90.8% success rate under a strict criterion that accepts a run only when every verdict and its supporting process are correct, and the result is consistent across the three LLMs tested (87.7%–94.6%). The ablation carries the central lesson: a ReAct loop with a Python interpreter, the plain combination of existing components, scores 0.0%; the specialized tools alone secure the computation but not the judgment; the expert knowledge base alone secures the judgment but not the computation. Standard-conformant assessment emerges only when both are present.

References

[1] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27730–27744.

[2] R. Zhang *et al.*, "LLaMA-Adapter: Efficient fine-tuning of large language models with zero-initialized attention," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024.

[3] D. Guo *et al.*, "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning," *Nature*, vol. 645, no. 8081, pp. 633–638, Sep. 2025.

[4] M. C. Tol and B. Sunar, "ZeroLeak: Automated side-channel patching in source code using LLMs," in *Proc. 29th Eur. Symp. Res. Comput. Secur.*, 2024, pp. 290–310.

[5] P. Liu *et al.*, "Exploring ChatGPT's capabilities on vulnerability management," in *Proc. 33rd USENIX Secur. Symp.*, 2024, pp. 811–828.

[6] S. Alnegheimish, L. Nguyen, L. Berti-Equille, and K. Veeramachaneni, "Can large language models be anomaly detectors for time series?" in *Proc. IEEE 11th Int. Conf. Data Sci. Adv. Anal.*, 2024, pp. 1–10.

[7] A. Mohamed, M. El Rashid, and K. Shaalan, "In-context learning in large language models (LLMs): Mechanisms, capabilities, and implications for advanced knowledge representation and reasoning," *IEEE Access*, vol. 13, pp. 95574–95593, 2025.

[8] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24824–24837.

[9] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023.

[10] M. Komeili, K. Shuster, and J. Weston, "Internet-augmented dialogue generation," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 8460–8478.

[11] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP models really able to solve simple math word problems?" in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 2080–2094.

[12] G. Wang *et al.*, "Voyager: An open-ended embodied agent with large language models," *Trans. Mach. Learn. Res.*, 2024.

[13] E. Karpas *et al.*, "MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," 2022, arXiv:2205.00445.

[14] N. Lawson, "Side-channel attacks on cryptographic software," *IEEE Secur. Privacy*, vol. 7, no. 6, pp. 65–68, Nov. 2009.

[15] S. Zhang, H. Wang, P. Qiu, Y. Lyu, H. Wang, and D. Wang, "SCAFinder: Formal verification of cache fine-grained features for side channel detection," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 8079–8093, 2024.

[16] Z. Qiao, Y. Liu, Y. Zhou, J. Ming, C. Jin, and H. Li, "Practical public template attacks on CRYSTALS-Dilithium with randomness leakages," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1–14, 2023.

[17] K. Qiao *et al.*, "A closer look at the belief propagation algorithm in side-channel attack on CCA-secure PQC KEM," *Sci. China Inf. Sci.*, vol. 67, no. 11, Nov. 2024, Art. no. 212302.

[18] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 19th Annu. Int. Cryptol. Conf. (CRYPTO)*, 1999, pp. 388–397.

[19] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smartcards," in *Proc. 1st Int. Workshop Cryptographic Hardware Embedded Syst. (CHES)*, 1999, pp. 144–157.

[20] *Security Requirements for Cryptographic Modules*, NIST FIPS PUB 140-3, 2019.

[21] *Information Technology—Security Techniques—Testing Methods for the Mitigation of Non-Invasive Attack Classes Against Cryptographic Modules*, ISO/IEC Standard 17825:2024, 2024.

[22] D. B. Roy, S. Bhasin, S. Guilley, A. Heuser, S. Patranabis, and D. Mukhopadhyay, "CC meets FIPS: A hybrid test methodology for first order side channel analysis," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 347–361, Mar. 2019.

[23] P. Schloyer, P. Knauer, B. Bauer, and D. Merli, "Automating side-channel testing for embedded systems: A continuous integration approach," in *Proc. 19th Int. Conf. Availability, Rel. Secur.*, 2024, pp. 1–6.

[24] I. Buhan, L. Batina, Y. Yarom, and P. Schaumont, "SoK: Design tools for side-channel-aware implementations," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2022, pp. 756–770.

[25] A. Geimer, M. Vergnolle, F. Recoules, L.-A. Daniel, S. Bardin, and C. Maurice, "A systematic evaluation of automated tools for side-channel vulnerabilities detection in cryptographic libraries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2023, pp. 1690–1704.

[26] U. Rioja, S. Paguada, L. Batina, and I. Armendariz, "The uncertainty of side-channel analysis: A way to leverage from heuristics," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 3, pp. 1–27, Jul. 2021.
[27] C. Whitnall and E. Oswald, "A critical analysis of ISO 17825 ('Testing methods for the mitigation of non-invasive attack classes against cryptographic modules')," in *Proc. 25th Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, 2019, pp. 256–284.
[28] F.-X. Standaert, "How (not) to use Welch's t-test in side-channel security evaluations," in *Proc. 17th Int. Conf. Smart Card Res. Adv. Appl. (CARDIS)*, 2018, pp. 65–79.
[29] M.-J. O. Saarinen, "WiP: Applicability of ISO standard side-channel leakage tests to NIST post-quantum cryptography," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, 2022, pp. 69–72.
[30] F. Yaman, "AgentSCA: Advanced physical side-channel analysis agent with large language models," Master's thesis, North Carolina State Univ., Raleigh, NC, USA, 2023.
[31] W. Waligóra, "Large language models in side-channel cryptanalysis," *Int. J. Electron. Telecommun.*, vol. 71, pp. 437–442, 2025.
[32] M. Nair, R. Sadhukhan, H. Pearce, D. Mukhopadhyay, and R. Karri, "Netlist Whisperer: AI and NLP fight circuit leakage!" in *Proc. Workshop Attacks Solutions Hardware Secur.*, 2023, pp. 83–92.
[33] D. Saha, H. Al Shaikh, S. Tarek, and F. Farahmandi, "Special Session: ThreatLens: LLM-guided threat modeling and test plan generation for hardware security verification," in *Proc. IEEE 43rd VLSI Test Symp.*, 2025, pp. 1–5.
[34] Z. Xu, Z. Wang, Y. Sun, and X. Wang, "Rethinking side-channel analysis: Automated discovery and analysis of side-channel leakage with LLM-assisted agents," 2026, arXiv:2605.17406.
[35] G. Deng *et al.*, "PentestGPT: Evaluating and harnessing large language models for automated penetration testing," in *Proc. 33rd USENIX Secur. Symp.*, 2024, pp. 847–864.
[36] C. Zhang, X. Lu, P. Cao, D. Gu, Z. Guo, and S. Xu, "A nonprofiled side-channel analysis based on variational lower bound related to mutual information," *Sci. China Inf. Sci.*, vol. 66, no. 1, Jan. 2023, Art. no. 112302.
[37] J. Trautmann, A. Beckers, L. Wouters, S. Wildermann, I. Verbauwhede, and J. Teich, "Semi-automatic locating of cryptographic operations in side-channel traces," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 1, pp. 345–366, 2022.
[38] W. Zhou *et al.*, "A prompt framework for LLM-based fully automated simple power analysis on cryptosystems," Cryptology ePrint Archive, Paper 2024/2069, 2024.
[39] S. Yao *et al.*, "Tree of thoughts: Deliberate problem solving with large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 11809–11822.
[40] Significant Gravitas, "AutoGPT," 2023, Accessed: Sep. 16, 2026. [Online]. Available: https://github.com/Significant-Gravitas/AutoGPT
[41] A. Osika, "GPT-Engineer," 2023, Accessed: Sep. 16, 2026. [Online]. Available: https://github.com/AntonOsika/gpt-engineer
[42] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Augmenting large language models with chemistry tools," *Nat. Mach. Intell.*, vol. 6, no. 5, pp. 525–535, May 2024.
[43] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol.*, 2023, pp. 1–22.
[44] O. Reparaz, J. Balasch, and I. Verbauwhede, "Dude, is my code constant time?" in *Proc. Design, Autom. Test Europe Conf. Exhib.*, 2017, pp. 1697–1702.