引用格式：宋富 . 密码实现安全形式化验证发展现状与展望[J].
前瞻科技, 2023, 2(1): 90-105; doi: 10.3981/j.issn.2097-0781.2023.01.007

密码实现安全形式化验证发展现状与展望

宋富
上海科技大学信息科学与技术学院，上海 201210

摘要 密码是保障网络安全和信息安全的核心技术和基础支撑，但密码实现本身面临多种安全威胁，形式化验证是严格证明密码实现安全性的重要技术手段。文章回顾了国内外在密码实现的功能正确性和内存安全性、时间侧信道安全性、功耗侧信道安全性方面的形式化验证技术发展现状和应用情况，分析了当前面向密码实现的形式化验证技术的不足、挑战和发展趋势。鉴于中国密码实现的安全性需求和相应形式化验证技术积累的不足，建议以研制构建高性能、高可信密码库为目标，以国家重大任务为牵引，通过顶层设计，强化自主创新，统一中间表示语言，研制高效率、高精度、自动化的形式化验证平台及安全编译优化工具链，最小化可信计算基，构造高效率、高可信密码实现通用库，促进产学研融合发展，进一步提高国家网络安全和信息安全的发展质量。

关键词 密码实现；安全与隐私；形式化验证；功能正确；内存安全；侧信道安全

从1976年开始，密码学的发展进入现代化，无论从深度和广度上都得到了空前的发展，密码的标准化工作和实际应用得到了各国政府、学术界和产业界的空前关注，世界各国和一些国际标准化组织高度重视密码标准的研究与制定，从而推动了密码学的研究与应用。例如，1997年美国启动国家标准技术研究院（National Institute of Standards and Technology, NIST）计划，2000年欧洲启动签名、完整性和加密的新欧洲方案（New European Schemes for Signatures, Integrity, and Encryption, NESSIE），2004年欧洲32所著名研究机构和企业启动欧洲杰出密码学网络（European Network of Excellence for Cryptology, ECRYPT）计划。2005年1月，中国经中央机构编制委员会批准成立国家密码管理局，此后国家密码管理局陆续公布了国密SM（ShangMi）系列密码标准，其中部分密码算法已经成为国际标准。2019年10月26日，中华人民共和国第十三届全国人民代表大会常务委员会第十四次会议审议通过了《中华人民共和国密码法》，自2020年1月1日起施行。密码法旨在规范密码应用和管理，促进密码事业发展，保障网络与信息安全，提升密码管理科学化、规范化、法治化水平，是中国密码领域的综合性、基础性法律。

收稿日期：2022-12-26；修回日期：2023-02-02

如图1所示，密码学可以提供的主要安全保障如下。

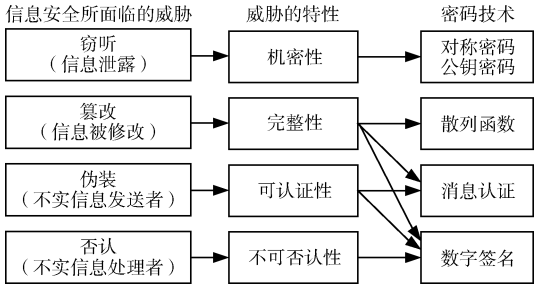


图1　密码对信息安全提供的安全保障

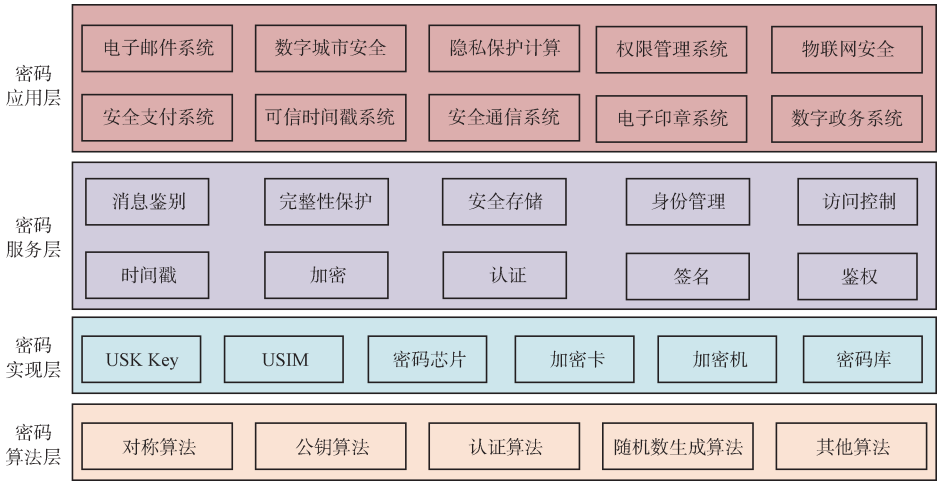
（1）机密性（Confidentiality）：保证敏感或机密信息在传输、接收、存储、使用等过程中不泄漏给未经授权者，甚至可以做到不暴露保密通信的事实。

（2）完整性（Integrity）：保证敏感或机密数据在传输、接收、存储、使用等过程中是完整的和未被篡改的，在被篡改的情况下能够发现篡改的事实或者篡改的位置。

（3）可认证性（Authentication）：也称真实性，保证实体（如人、进程或系统）身份或信息、信息来源的真实性。

（4）不可否认性（Non-repudiation）：保证信息系统的操作者或信息的处理者不能否认其行为或者处理结果。

鉴于上述4个重要的安全保障，如图2所示，密码在各个领域得到了广泛的应用，成为网络安全与信息安全的核心技术和基础支撑，是实现内生安全的基石，是解决安全问题最有效、最可靠、最经济的手段。根据恒州博智《2020—2026年全球和中国商用密码市场现状及未来发展趋势》研究报告，2019年全球商用密码市场规模达到了250.4亿美元，预计2026年将达到864.1亿美元，年复合增长率为18.79%。根据中金企信《2022—2028年中国商用密码行业市场全景调研分析及投资可行性研究预测报告》，近年来中国商用密码行业规模不断扩大，产业规模整体呈上升趋势；2020年在新冠疫情流行的客观环境下，中国商用密码产业仍取得高速发展，总体规模达到466.0亿元，较2019年增长33.14%，预计2023年商用密码行业规模有望达到937.5亿元。



USB Key，Universal Serial Bus Key，通用串行总线接口的加密锁；USIM，Universal Subscriber Identity Module，全球用户识别卡。

图2　密码应用技术体系

1　密码实现面临的安全威胁

由于密码在网络安全和信息安全中的基础性和重要性，密码本身的安全性变得尤为重要。在古典密码学阶段，密码学专家常凭直觉和信念进行密码设计和分析，而不是依靠推理证明。1945

—1975年近代密码学阶段，香农在1949年发表的论文《保密系统的信息理论》为对称密码学建立了理论基础，密码学专家开始采用信息论研究密码算法安全性。在现代密码学研究中，密码学专家追求计算安全性和可证明安全性，关注点从密码学的基本构件转移到用信息论和计算复杂性理论研究密码学，试图在某一理论模型下证明密码算法的安全性。现代密码算法在成为标准之前，除了密码算法设计者会进行理论安全性的分析和证明之外，也会公开密码算法草案，由国际同行专家进行密码分析，尝试通过线性密码分析、差分密码分析、代数密码分析等传统攻击技术有效地评估密码算法的安全性。基于混合整数线性规划（Mixed Integer Linear Program, MILP）求解[1-2]和基于交互式证明的工具[3]出现为计算机辅助证明密码算法的安全性提供了技术支撑和工具保障。

但是在理论上计算安全或者可证明安全的密码算法，在密码实现层仍可能存在安全缺陷，导致密码实现在实际应用中无法提供安全保障。密码实现的安全问题主要包括功能正确性、内存安全性和侧信道安全性。密码实现的功能正确性和内存安全性问题来源于密码算法不正确的程序或硬件实现，特别是密码算法涉及大整数运算、非线性运算、汇编级手动优化时极易出现此类安全问题。例如，Bernstein等[4]设计实现用于高性能、高安全签名的椭圆曲线加密算法存在功能正确性缺陷。由于该缺陷触发的概率极低，大量软件测试都无法发现，导致该缺陷长时间存在。侧信道安全性是指密码实现实际运行时产生的侧信道信息与隐私数据（如密钥）相关性导致的安全缺陷，攻击者可以利用侧信道信息与密钥的相关性快速、高效地破解密钥。如图3所示，侧信道信息包括电磁、时间、声音、温度和功耗等。自1996年Kocher[5]首次开创性地提出时间侧信道攻击并破解了Diffie-Hellman、RSA（Rivest-Shamir-Adleman）等密码算法实现之后，

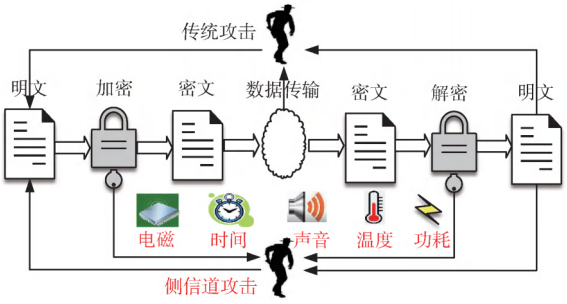


图3　密码实现侧信道攻击

Kocher等[6-8]分别利用功耗、电磁侧信道破解数据加密标准（Data Encryption Standard, DES）、高级加密标准（Advanced Encryption Standard, AES）等密码算法；Brumley和Tuveri[9]甚至利用时间侧信道远程破解服务器端著名开源安全套接字层的密码库（Open Secure Sockets Layer, OpenSSL）中的椭圆曲线签名。近几年，OpenSSL时间侧信道漏洞DROWN（Decrypting RSA with Obsolete and Weakened eNcryption）[10]影响了1100多万个网站的安全，中央处理器（Central Processing Unit, CPU）指令重排序和预测执行造成的时间侧信道漏洞Meltdown[11]和Spectre[12]几乎影响了全球20年的内所有CPU。

笔者统计OpenSSL在2016年1月1日至2019年5月1日披露的24个通用漏洞（Common Vulnerabilities and Exposures, CVE）发现，其中有10个功能正确性漏洞，5个内存安全漏洞和8个侧信道安全漏洞。

2　国内外密码实现安全性形式化验证发展现状

当前国内外研究主要聚焦于密码实现功能正确性和内存安全性、时间侧信道安全性、功耗侧信道安全性的形式化验证，而电磁、声音和温度侧信道信息泄漏难于形式化建模，目前尚无相关形式化验证工作，因此本文仅总结国内外密码实现功能正确性和内存安全性、时间侧信道安全性、功耗侧信道安全性的形式化验证发展现状。

2.1 密码实现功能正确性和内存安全性形式化验证

密码实现面临的功能正确性和内存安全性问题得到了国内外相关研究机构和企业的高度重视，多个研究团队持续探索研究，提出了各类形式化验证技术路线，并对部分国际主流密码实现进行了示范应用研究。表1总结了密码实现功能正确性和内存安全性形式化验证的代表性工作及相关信息，工具按照最新发表论文的时间排序。第3列中●表示支持，○表示不支持，◉表示部分支持。第5列描述对应工具的自动化程度中●表示自动验证，○表示交互式验证，◉表示部分自动验证和部分交互式验证。第6列中CryptoLine和Vale是类汇编的特定领域命令式程序设计语言，WhyML、Gallina和F*是特定领域函数式程序设计语言，Jasmin是支持高级语言和汇编级语言特性的特定领域命令式程序设计语言。

表1　密码实现功能正确性和内存安全性形式化验证的研究进展

验证工具和文献	年份	内存安全	验证技术	自动化程度	输入语言	目标语言	应用
Frama-C[13]	2013	●	抽象解释 演绎推理	◉	C	C	RSA-OEAP, MEE-CBC
gfverif[14]	2015	○	代数系统	●	C	C	X25519
Cryptol+SAW[15]	2016	●	程序等价 静态符号执行	◉	C, Java	C, Java	Salsa20, AES, ZUC, FFS, ECDSA, SHA-3
Fiat Crypto[16]	2019	●	定理证明	○	Gallina	C	Curve25519, P256
EverCrypt[17]	2020	●	演绎推理 定理证明	◉	F*, Vale	C, ASM, WASM	HACL*库和ValeCrypt
Why3[18-19]	2021	◉	演绎推理	◉	WhyML	OCaml, ASM	BGW, X25519
VST[20-23]	2021	●	定理证明	○	Gallina	C	DRBG, SHA-256, SHA-2, X25519
Jasmin[24-27]	2021	●	抽象解释 演绎推理 定理证明	◉	Jasmin	C, ASM	ChaCha20, Poly1305, SHA-3
CryptoLine[28-30]	2022	●	演绎推理 代数系统	●	CryptoLine	C, ASM	NaCl, OpenSSL等库函数

上述形式验证工具的主要目的是证明密码实现的功能正确性和内存安全性。例如，CryptoLine已被应用于验证开源项目OpenSSL、boringSSL、NaCl （Networking and Cryptography Library）、wolfSSL和bitcoin中多条椭圆曲线加密算法相关函数的C实现的功能正确性和内存安全性，以及后量子密码（Post-Quantum Cryptography, PQC）方案Kyber、SABER和NTRU中数论变换（Number Theoretic Transform, NTT）的汇编实现的正确性；EverCrypt构建了可证明正确的密码实现库，部分密码实现集成到Windows和Linux操作系统、Azure云和Firefox浏览器；Fiat Crypto证明的密码实现集成到谷歌的安全套接字层BoringSSL库。除此之外，形式验证工具也可以发现密码实现中的缺陷和漏洞，如CryptoLine揭露了NaCl密码库中Curve25519椭圆曲线密码实现的1个缺陷、NIST P-521椭圆曲线加密实现中1个缺陷和2个错误假设。

2.2 密码实现时间侧信道安全性形式化验证

针对密码实现的时间侧信道安全问题，目前有两种不同的解决策略，分别是常量时间实现和平衡时间实现。前者要求密码实现中所有可能造成执行时间差异的逻辑跟密钥无关，需要对密码实现进行复杂的修改，给密码实现的功能正确性

和内存安全性带来巨大挑战；后者仅要求密码实现在输入不同密钥时执行时间差异小到可忽略，因此相对容易实现，但无法提供绝对的时间侧信道安全保证。无论采用哪种解决策略，都需要应用形式化验证严格保证密码实现的时间侧信道安全性。表2总结了密码实现时间侧信道安全性形式化验证的代表性工作及相关信息，工具按照最新发表论文时间排序。第3列中①表示分支不平衡引起的执行时间差异；②表示内存访问高速缓存命中（Cache Hit）和高速缓存缺失（Cache Miss）引起的执行时间差异；③表示浮点数指令操作数范围不同引起的执行时间差异；④表示CPU分支预测引起的执行时间差异；⑤表示CPU指令重排序引起的执行时间差异；⑥表示编译优化造成的执

表2　密码实现时间侧信道安全性形式化验证的研究进展

验证工具和文献	年份	泄漏模型	验证技术	验证精度	输入语言	目标语言	应用
CacheAudit[31]	2013	①②	抽象解释	◉	二进制	二进制	AES, eSTREAM
ct-verif[32]	2016	①②③	交叉积	●	C	Boogie	Nacl, OpenSSL, BoringSSL
Themis[33]	2017	①	污点分析 自组合	○	Java	Java	DARPA STAC测试集
Blazer[34]	2017	①	污点分析 复杂度比较	○	Java	Java	DARPA STAC测试集
IFC-CEGAR/BMC[35]	2018	①②	污点分析 惰性自组合	●	LLVM IR	C	—
SC Eliminator[36]	2018	①②	类型系统 自动修复	◉	LLVM IR	C	AES, DES, LBlock, PRESET等
FaCT[37]	2019	①②③	类型系统 自动修复	◉	FaCT	LLVM IR	Poly1305, XSalsa20, MEE-CBC, Curve25519
CT-Wasm[38]	2019	①②③	类型系统	◉	WASM	WASM	Salsa20, SHA-256, TweetNaCl
AISE[39]	2019	②④	抽象解释	◉	LLVM IR	C	hpn-ssh, LibTomCrypt, OpenSSL,Tegra
EverCrypt[17]	2020	①②③	污点分析 类型系统	◉	Low*, Vale	C, ASM	HACL*库和ValeCrypt
SpecuSym[40]	2020	④	动态符号执行	○	LLVM IR	C	hpn-ssh, LibTomCrypt,
KleeSpectre[41]	2020	④	动态符合执行	○	LLVM IR	C	OpenSSL,Tegra
BINSEC/REL[42]	2020	①②	关系符号执行	○	汇编	二进制	HACL*, OpenSSL, BearSSL
SPECTECTOR[43]	2020	④	静态符号执行	◉	汇编	汇编	—
Pitchfork[44]	2020	④⑤	污点分析 静态符号执行	◉	汇编	汇编	Poly1305, XSalsa20, MEE-CBC, Curve25519
Oo7[45]	2020	①②④⑤	污点分析 自动修复	○	二进制	二进制	Litmus Tests, SPECint
Blade[46]	2021	④⑤	类型系统 自动修复	◉	WASM	WASM, C	Salsa20, SHA-256, ChaCha20, Poly1305, Curve25519
Binsec/Haunted[47]	2021	①②④⑤	关系静态 静态符号执行	○	汇编	二进制	Libsodium, OpenSSL
Jasmin[24-27,48]	2022	①②④⑤⑥	定理证明	●	Jasmin	C, 汇编	ChaCha20, Poly1305, keccak1600等
DeJITLeak[49]	2022	①⑥	类型系统 自动修复	◉	Java 字节码	Java 字节码	DARPA STAC测试集

行时间差异。第5列中◉表示可靠但不完备；○表示既不可靠又不完备；●表示可靠且完备。

由表2可见，大部分验证工具已被应用于证明多个开源密码实现的时间侧信道安全性。除此之外，形式验证工具也可以检测出密码实现中难以发现的时间侧信道漏洞。例如，LibTomCrypt密码库的blowfish对称密码实现和chacha20流密码实现，OpenSSL密码库中DES密码实现中的缓存时间侧信道漏洞；GCC和LLVM编译器编译优化引入的时间侧信道漏洞；Java虚拟机HotSpot运行时编译优化引入的时间侧信道漏洞。

2.3 密码实现功耗侧信道安全性形式化验证

针对密码实现的功耗侧信道安全问题，目前主流的解决策略是一种基于密钥共享（Secret Sharing）的掩码（Masking），顾名思义，就是将密钥拆分成指定数量的共享，然后加密和解密在密钥共享上进行运算。为了保证在密钥共享上进行运算的密码实现的功耗侧信道安全性，国内外专家提出了多种形式化验证技术。

表3总结了密码实现功耗侧信道安全性形式化验证的代表性工作及相关信息，工具按照最新发表论文时间排序。第3列中①表示探测安全

表3　密码实现功耗侧信道安全性形式化验证的研究进展

验证工具和文献	年份	安全模型	阶数	可组合验证	验证技术	验证精度	输入语言	目标语言	应用
SC Sniffer[50-52]	2014	①⑦	1,2	N	约束求解 自动修复	●	布尔电路	C	Keccak, AES-Sbox
EasyCrypt[53]	2015	①⑥	≥1	N	证明系统	◉	EasyCrypt	EasyCrypt	Keccak, AES, AES-Sbox
maskComp[54]	2016	②③	≥1	Y	证明系统 自动修复	◉	C	C	AES, Keccak, Simon, Speck
BYT[55]	2017	①	≥1	Y	约束求解 自动修复	●	布尔电路	C	Keccak, AES-Sbox
CheckMasks[56]	2018	②③	≥1	N	初等变换	◉	Lisp	Lisp	布尔算术互转函数
Rebecca[57]	2018	①⑤	≥1	N	傅里叶分析 约束求解	◉	布尔电路	Verilog	Keccak/AES-Sbox
SC Infer[58-59]	2019	①⑦	1	N	类型系统约束 求解混合验证	●	布尔电路	C	Keccak, AES-Sbox
SC Sniffer2[60]	2019	①⑥	1	N	类型系统 自动修复	◉	布尔电路	C	Keccak, AES-Sbox
maskVerif[61]	2019	①②③⑤⑥	≥1	N	证明系统	◉	布尔电路	Verilog	Keccak/AES-Sbox
tightPROVE+[62-63]	2020	①	≥1	Y	线性代数 自动修复	◉	布尔电路	Usuba	NIST 2nd Round 轻量级加密算法
SILVER[64]	2020	①②③④⑤	≥1	N	二元决策图	●	布尔电路	Verilog	AES/PREENT/ PRINCE-Sbox
VRAPS[65]	2020	⑧	≥1	Y	类型系统	◉	布尔电路	Sage	AES
HOME[66]	2021	①	≥1	N	类型系统约束 求解混合验证	●	算术电路	C	Keccak, AES
fullverif[67]	2021	④⑤	≥1	Y	依赖分析	●	布尔电路	Verilog	Sbox
QMV_{ERIF}[68]	2022	①⑥⑦	1	Y	类型系统约束 求解混合验证	●	算术电路	C	Keccak, AES-Sbox
IronMask[69]	2022	②③④⑤⑥⑧	≥1	N	高斯消元	◉	布尔电路	Sage	乘法器

（Probing Security），要求任意d个中间变量值的联合概率分布独立于密钥，d是安全模型阶数；②表示不干涉（Non-interference），要求任意d个中间变量值的联合概率分布可以通过包含每个输入密钥最多d个安全共享的变量组合进行模拟；③表示强不干涉（Strong Non-interference），要求任意d_1个中间变量和$d-d_1$个输出变量的值的联合概率分布，可以通过包含每个输入密钥最多d_1个安全共享的变量组合进行模拟；④表示探测隔离不干涉（Probe-isolating Non-interference），要求任意d_1个中间变量和每个输出变量的任意$d-d_1$个安全共享的值的联合概率分布可以通过包含每个输入密钥最多d_1个对应安全共享的变量组合进行模拟；⑤表示毛刺（Glitches），指1个中间变量同时关联该中间变量计算所依赖的所有输入或寄存器输出；⑥表示转变（Transitions），指1个中间变量同时关联该变量被修改前后的值；⑦表示定量掩码强度，指当探测安全不成立时破解密钥所需功耗信息量；⑧表示随机探测安全（Random Probing Security），指安全阶数d等于所有非零泄漏概率的中间变量。安全模型①、②、③、④的要求按顺序越来越强，因此，探测隔离不干涉的密码实现是强不干涉的。强不干涉的密码实现是不干涉，不干涉的密码实现是探测安全，但反之不成立。安全模型①、②、③、④、⑧都可以扩展加入毛刺和转变，安全模型①、②、③、④的毛刺和转变扩展的要求依次变强。第4列的安全阶数越高，密码实现抵抗功耗侧信道的攻击的能力越强，但密码实现的计算量和随机变量数就越大，因此执行性能越低。第5列中N表示不支持，Y表示支持，非组合验证技术需要考虑密码实现中跨模块中间变量组合的值的联合概率分布；可组合验证技术通过验证密码实现中各模块内中间变量组合和模块之间可组合性来完成密码实现的安全性验证。第7列中◉表示可靠但不完备，●表示可靠且完备。值得注意的是，此处的完备性和可靠性是针对给定安全模型而言的，验证其他安全模型时不一定可靠或完备。第8列中布尔电路表示变量只能取0和1的布尔变量，运算只能是逻辑运算；算术电路表示变量可以是固定位宽的正整数，运算可以有逻辑运算和算术运算；EasyCrypt是特定领域程序语言；Lisp是一种通用函数式程序语言。

目前，大部分功耗侧信道安全性形式验证工具的对象是AES等对称加密算法和Keccak哈希算法的实现，已经证明多个密码算法掩码实现的功耗侧信道安全性。除此之外，已发表的AES和Keccak的掩码实现中被发现了多个功耗侧信道漏洞。

2.4 国内外研究现状对比

从密码实现功能正确性和内存安全性、时间侧信道安全性形式化验证、功耗侧信道安全性形式化验证3个方面对比国内外研究现状。

1）密码实现功能正确性和内存安全性

CryptoLine工具由中国台湾中央研究院和深圳大学团队主导研制，采用基于约束求解的演绎推理和代数系统进行验证，目前处于验证开源密码实现部分函数的阶段，其优点是自动化程度高，不足之处是可验证密码实现的规模小而无法对完整密码实现进行形式化验证。除CryptoLine之外，其他所有工具由欧美国家的研究机构和企业主导。起步时间相对较早，采用抽象解释、演绎推理、定理证明等多种技术方案，具有扎实的技术积累，研制的形式化验证工具已经应用于验证开源密码实现的函数。相对比较成熟的EverCrypt和Jasmin项目经过了5年以上发展，构筑形成了相对成熟的可验证安全密码实现库，具有多个领域的示范应用。

2）密码实现时间侧信道安全性形式化验证

DeJITLeak工具由上海科技大学团队主导研制，针对Java程序运行时编译优化引起的时间侧信道漏洞自动检测和修复技术，采用基于类型推导的信息流分析技术定位潜在时间侧信道漏洞、

通过禁止潜在时间侧信道漏洞相关代码的优化来避免时间侧信道漏洞。除DeJITLeak之外，其他所有工具均由欧美国家的研究机构和企业主导，支持更多高级程序语言和汇编级密码实现的形式化验证，采用抽象解释、交叉积、污点分析、自组合、复杂度比较、惰性自组合、类型系统、符号执行等不同验证精度和验证效率的技术方案，考虑了更加全面和复杂的时间差异源，拥有配套安全编译器，部分工具甚至兼顾了密码实现功能正确性和内存安全性的形式化验证。

3）密码实现功耗侧信道安全性形式化验证

除SCInfer、HOME和QMV_{ERIF}工具由上海科技大学团队主导研制外，其他所有工具均由欧美国家的研究机构和企业主导。虽然SCInfer、HOME和QMV_{ERIF}采用类型推导和约束求解的混合验证技术，在验证效率和验证精度上取得了国际领先的水平，但是目前尚不支持毛刺和转变导致的功耗侧信道安全问题，也不能对高阶功耗侧信道安全性进行组合验证。国外工具已经考虑更加复杂的功耗侧信道安全性问题，支持高阶功耗侧信道安全性的组合验证，并具备功耗侧信道安全密码实现的自动生成能力。

综上所述，得益于欧洲研究委员会（European Research Council, ERC）和美国国防部高级研究计划局（Defense Advanced Research Projects Agency, DARPA）等政府机构的相关顶层研究计划、经费投入和资助，欧美学者组建了多个研究团队，取得了出色的研究成果，对欧美密码算法的实现进行多维度的安全性验证。技术发展和成果积累进一步推动了产业化应用和密码实现安全认证新标准的建立。法国CryptoExperts公司和Cryspen公司均在研制密码实现安全性形式化验证技术和工具，并构建可验证、高性能、高安全密码程序进行产业化推广应用。美国国家标准技术研究院正在讨论制定密码实现功耗侧信道安全缓解机制和认证标准，其中侧信道安全性的形式模型和形式化验证工具是认证标准中的讨论重点和首要技术手段。

虽然中国在部分密码实现安全性问题上取得了突破，但是在研究团队规模和数量，政府机构整体规划和投入，技术水平和积累，形式化验证工具的研制，可验证、高性能、高安全密码程序的构筑和产业化应用，密码安全认证新标准的建立等方面都落后于欧美发达国家，特别是密码安全认证标准还停留在依靠传统攻击技术测试密码实现的安全性。据笔者调研，目前没有任何形式化验证工具应用于国密SM系列算法实现的形式化验证。鉴于国密SM系列算法在中国的特殊性和重要性，密码实现安全性问题对中国的网络安全和信息安全构成重大隐患。

3 发展趋势

密码实现安全性形式化验证技术发展趋势可以总结为以下5点。

1）最小化可信计算基

虽然形式化验证技术，一方面可以在理论上证明密码实现安全性，但是形式化验证的算法或实现的程序代码可能存在缺陷；另一方面大部分形式化验证工具依赖的定理证明器或约束求解器也可能存在缺陷。这些缺陷可能会导致形式化验证工具无法发现密码实现的缺陷。因此，最近的研究工作开始关注形式化验证算法、定理证明器或约束求解器的形式化验证，并设计安全编译优化工具链，从而减小可信计算基。

2）可验证、高效率密码实现

密码实现的源代码验证比手动优化底层实现的验证更加容易，而手动优化底层实现的性能比源代码实现的更优。早期形式化验证工具针对密码源代码程序实现，因此经过形式化验证的密码实现的性能劣于手动优化底层实现。针对这个不

足，部分研究团队开始研制面向手动优化汇编实现的形式化验证工具和高效率汇编程序的编译优化生成。虽然这需要更复杂的验证技术方法和更多的验证工作量，目前部分经形式化验证的密码实现的性能可以跟密码专家手动优化的汇编实现相媲美，甚至性能更优于密码专家手动优化的汇编实现。这说明形式化验证在提高密码实现安全性的同时，不会牺牲密码实现的太多性能。

3）更高的自动化程度

密码算法涉及复杂的大整数、非线性运算，特别是手动优化的密码实现，对形式化验证的自动化带来巨大的挑战。因此，当前大部分研究团队采用交互式证明的方式来开展验证工作，但交互式验证工具的使用对于密码学专家而言非常困难。虽然依赖自动定理证明器和计算代数约束求解器的完全自动化验证技术有了一定的研究进展，但由于可扩展性问题，目前只能验证部分密码算法中的关键函数实现。对于如何自动化验证更大规模，甚至完整的密码实现是未来亟须解决的技术挑战。

4）高精度形式化验证与自动修复结合

形式化验证，一方面只能证明密码实现的安全性，但当密码实现存在漏洞时，形式化验证本身不能修复漏洞，因此最近的研究工作开始探索通过程序指令变化的方法自动修复形式化验证识别的潜在漏洞。另一方面，与自动修复结合的形式化验证技术普遍是轻量级的，均是可靠但不完备的形式化验证技术。此类技术会存在大量的误报，对误报的漏洞相关代码进行修改导致密码实现的性能下降，也对修改后密码实现其他安全性的形式化验证带来极大的挑战。未来有望将高精度形式化验证技术与自动修复结合，通过高精度形式化验证技术减少误报，从而降低漏洞自动修复带来的性能损失。

5）安全模型和泄漏模型更全面

早期考虑的时间侧信道安全性形式化验证的泄漏模型主要关注程序分支不平衡引起的执行时间差异和程序内存访问高速缓存命中和高速缓存缺失引起的执行时间差异等。近几年，时间侧信道泄漏模型的聚集点开始转移到处理器引起的执行时间差异源和编译优化造成的时间差异。功耗侧信道的探测安全、不干涉、强不干涉和探测隔离不干涉4个标准安全模型对实际物理设备执行密码实现时的功耗侧信道信息进行了抽象，忽略了在实际物理设备运行密码实现时存在毛刺和转变等物理缺陷，导致探测安全、不干涉、强不干涉或探测隔离不干涉的密码实现在实践中仍然可能存在功耗侧信道信息泄漏。因此，这4个标准安全模型的毛刺和转变扩展开始得到研究人员的关注。随着基于机器学习和深度学习的功耗侧信道攻击的出现，4个标准安全模型的毛刺和转变扩展也无法真实刻画功耗侧信道的安全模型，随机探测安全及其毛刺扩展安全模型也随之被提出。因此，未来更加符合实际侧信道安全的泄漏模型和安全模型及其验证技术将是研究的热点之一。

4 中国密码实现安全性形式化验证发展面临的挑战及发展建议

4.1 面临的挑战

中国密码实现安全性形式化验证发展主要面临以下4个重大挑战。

1）缺少国家顶层设计和发展计划

目前，密码实现安全性问题没有得到中国政府机构和行业的足够重视，未出台任何相关建议和指导性文件，也没有制定顶层的设计和发展规划。而美国国防部高级研究计划局在2014年制订了相关研究计划，提供5300万美元资助了一批高校和企业开展研究工作，构建了标准测试集，培养了多个研究团队、研制了不同技术方案的密码实现安全性形式化验证工具，并初步构建了可验

证、高安全、高效率密码库。缺少国家顶层设计和发展计划都不利于中国密码实现安全性形式化验证的技术研究和产业化应用。

2）密码安全认证标准滞后

中国密码安全认证标准采用传统已知攻击测试的方法，没有建议或要求密码实现通过形式化验证的方式进行安全性评估，而美国国家标准技术研究院已经在规划推进通过形式化验证进行密码实现安全认证标准的更新。虽然国家信息安全测评认证中心主持的GB/T 18336《信息技术 安全技术 信息技术安全性评估准则》（等同于ISO/IEC 15408，通常简称通用准则CC）评估标准高等级认证已经强制要求采用形式化技术，但该标准不适用于密码实现安全性的认证和行业要求。

3）密码实现研发和应用行业不重视

企业以盈利为目的，重视产品推向市场的速度和产品用户体验，而提高密码实现安全性和研发形式化验证工具本身难以为企业带来经济效益，反而会延误产品上市速度。在密码实现安全认证标准不要求形式化验证时，网络安全和信息安全事故发生概率小、代价低的现实情况下，企业以使用开源密码库为主，没有动力投入研发力量推动密码实现安全性形式化验证技术的发展和可验证、高安全、高效率密码库的研制。

4）研究团队少又小

密码学和形式化验证，一方面本身属于技术门槛高的研究领域，相比人工智能等热门行业，研究和从业人员数量要少很多，而密码实现形式化验证更是小众的交叉研究领域，需要具备密码学和形式化验证的技术知识才能开展研究工作，导致研究和从业人员更是凤毛麟角；另一方面大部分学生以“好就业”“高薪酬”为导向选择学习和研究的领域，造成后备青年研究人才培养的不足。

4.2 发展建议

根据中国密码实现的安全性需求和相应形式化验证技术积累的不足，提出以下4点发展建议。

1）加强顶层设计，有计划地持续推动密码实现的安全性形式化验证发展

针对密码实现安全的重要性、密码实现威胁的多样性和复杂性，当前研究各自聚焦密码实现的部分安全性问题。因此，建议通过顶层设计，规划战略研究计划，分析密码实现各安全性问题形式化验证的技术挑战，根据密码算法软硬件实现的特点，设计有效、可扩展、可集成的泄漏模型和安全模型，更新密码安全认证标准，长期持续研制高精度、高效率的形式化验证技术，夯实密码实现安全性形式化验证基础理论，为设计实现高效率、高可信密码库提供技术和工具支撑。

2）统一中间表示语言和形式化验证平台

当前密码实现形式化验证工具输入语言多样且不兼容，无法利用已有工具对验证密码实现开展多维度安全性形式化验证。亟须设计面向密码软硬件实现的统一中间表示语言，模块化可配置的泄漏模型和安全模型，在此基础上设计开发相关安全性形式化验证技术，构建统一的密码实现安全性形式化验证平台，提供一站式安全性形式化验证服务，避免重复造车又无法集成。

3）强化自主可控、安全可信形式化验证工具，构造高安全、高效率密码库

在当前严苛的国际环境下，基础软件自主可控、安全可信是国家网络空间安全的重中之重，而密码是网络安全和信息安全的核心技术和基础支撑。为保障国家网络空间安全，中国应以国家科技专项为牵引，加强人力和物力投入，梳理密码实现安全性形式化验证的关键核心技术，研制自主可控、安全可信的密码实现安全性形式化验证基础平台，在此基础上构建自主可控、安全可信的密码软硬件库以及安全编译优化工具链，最小化可信计算基，特别是国密SM系列密码标准实现库。

4）促进产学研融合发展

以国家重大任务为牵引、国家商用密码标准为抓手，实施科研院所、高校、企业和商用密码检测机构联合攻关，促进密码实现安全及其形式化验证的研究、开发和应用的综合发展，快速提高中国密码实现基础软件的安全性。完善相关人才队伍建设和培养的体制机制，打造一支具备程序语言、密码学、形式化验证专业知识的复合型人才梯队。

5 结束语

网络安全和信息化是事关国家安全和国家发展、事关广大人民群众工作生活的重大战略问题。密码是保障网络安全与信息安全的核心技术和基础支撑。作为保障密码实现安全的形式化验证已经在多个方面取得了重要进展，相关密码实现也在工业界进行了示范应用，取得了显著效果。虽然中国在少数几个关键技术上也取得了突破性进展，但仍有许多亟待突破的关键技术。针对密码实现安全性形式化验证的急迫性和重要性，建议以研制构建自主可控、安全可信形式化验证工具和高性能S3L高可信密码库为目标，加强顶层设计，强化自主创新，设计统一中间表示语言、形式化验证平台和安全编译优化工具链，促进产学研融合发展，提升中国自主密码实现安全水平，为国家网络安全和信息安全保驾护航。

参考文献

[1] Mouha N, Wang Q J, Gu D W, et al. Differential and linear cryptanalysis using mixed-integer linear programming[C]//Yung M, Liu P, Lin D. Proceedings of 4th International Conference on Information Security and Cryptology. Berlin, Heidelberg: Springer, 2012: 57-76.

[2] Sun S W, Hu L, Wang P, et al. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers[C]//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. Berlin, Heidelberg: Springer, 2014: 158-178.

[3] Barthe G, Dupressoir F, Grégoire B, et al. EasyCrypt: A tutorial[C]//Aldini A, Lopez J, Martinelli F. Proceedings of the International School on Foundations of Security Analysis and Design VII , FOSAD 2012/2013. Cham: Springer, 2014: 146-166.

[4] Bernstein D J, Duif N, Lange T, et al. High-speed high-security signatures[J]. Journal of Cryptographic Engineering, 2012, 2(2): 77-89.

[5] Kocher P C. Timing attacks on implementations of diffie-Hellman, RSA, DSS, and other systems[C]//Koblitz N. Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO'96. Berlin, Heidelberg: Springer, 1996: 104-113.

[6] Kocher P, Jaffe J, Jun B. Differential power analysis[C]//Stern J. Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology—EUKOCRYPT'99. Berlin, Heidelberg: Springer, 1999: 388-397.

[7] Quisquater J J, Samyde D. ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards [C]//Attali I, Jensen T. Proceedings of the International Conference on the Smart Card Programming and Security. Berlin, Heidelberg: Springer, 2001: 200-210.

[8] Mangard S, Pramstaller N, Oswald E. Successfully attacking masked AES hardware implementations[C]//Rao J R, Sunar B. Proceedings of the 7th International Workshop on Cryptographic Hardware and Embedded Systems —CHES 2005. Berlin, Heidelberg: Springer, 2005: 157-171.

[9] Brumley B B, Tuveri N. Remote timing attacks are still

practical[C]//Atluri V, Diaz C. Proceedings of the 16th European Symposium on Research in Computer Security—ESORICS 2011. Berlin, Heidelberg: Springer, 2011: 355-371.

[10] Aviram N, Schinzel S, Somorovsky J, et al. DROWN: Breaking TLS using SSLv2[C]//Proceedings of the USENIX Security Symposium. Berkeley: USENIX Association, 2016: 689-706.

[11] Lipp M, Schwarz M, Gruss D, et al. Meltdown: Reading kernel memory from user space[C]//Proceedings of the USENIX Security Symposium. Berkeley: USENIX Association, 2018: 973-990.

[12] Kocher P, Horn J, Fogh A, et al. Spectre attacks: Exploiting speculative execution[C]//Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2019: 1-19.

[13] Almeida J B, Barbosa M, Barthe G, et al. Certified computer-aided cryptography: Efficient provably secure machine code from high-level implementations[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. New York: ACM Press, 2013: 1217-1230.

[14] Bernstein D J, Schwabe P. Gfverif: Fast and easy verification of finite-field arithmetic[EB/OL]. [2022-12-14]. http://gfverif.cryptojedi.org.

[15] Tomb A. Automated verification of real-world cryptographic implementations[J]. IEEE Security & Privacy, 2016, 14(6): 26-33.

[16] Erbsen A, Philipoom J, Gross J, et al. Simple high-level code for cryptographic arithmetic-with proofs, without compromises[C]//Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2019: 1202-1219.

[17] Protzenko J, Parno B, Fromherz A, et al. EverCrypt: A fast, verified, cross-platform cryptographic provider[C]//Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2020: 983-1002.

[18] Filliâtre J C, Paskevich A. Why3—Where programs meet provers[C]//Felleisen M, Gardner P. Proceedings of the 22nd European Symposium on Programming. Berlin, Heidelberg: Springer, 2013: 125-128.

[19] Schoolderman M, Moerman J, Smetsers S, et al. Efficient verification of optimized code[C]//Dutle A, Moscato Mm, Titolo L, et al. Proceedings of the 13th International Symposium on NASA Formal Methods. Cham: Springer, 2021: 304-321.

[20] Appel A W. Verified software toolchain[C]//Goodloe A E, Person S. Proceedings of the 4th International Symposium on NASA Formal Methods. Berlin, Heidelberg: Springer, 2012, doi: 10.1007/978-3-642-28891-3_2.

[21] Beringer L, Petcher A, Ye K Q, et al. Verified correctness and security of OpenSSL HMAC[C]//Proceedings of the USENIX Security Symposium. Berkeley: USENIX Association, 2015: 207-221.

[22] Ye K Q, Green M, Sanguansin N, et al. Verified correctness and security of mbedTLS HMAC-DRBG[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2017: 2007-2020.

[23] Schwabe P, Viguier B, Weerwag T, et al. A Coq proof of the correctness of X25519 in TweetNaCl[C]//Proceedings of the 2021 IEEE 34th Computer Security Foundations Symposium. Piscataway: IEEE Press, 2021, doi: 10.1109/CSF51468.2021.00023.

[24] Almeida J B, Barbosa M, Barthe G, et al. Jasmin: High-assurance and high-speed cryptography[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2017: 1807-1823.

[25] Almeida J B, Baritel-Ruet C, Barbosa M, et al. Machine-checked proofs for cryptographic standards: Indifferentiability of sponge and secure high-assurance implementations of SHA-3[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 1607-1622.

[26] Almeida J B, Barbosa M, Barthe G, et al. The last mile: High-assurance and high-speed cryptographic implementations[C]//Proceedings of the 2020 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2020: 965-982.

[27] Barthe G, Cauligi S, Grégoire B, et al. High-assurance cryptography in the spectre era[C]//Proceedings of the 2021 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2021: 1884-1901.

[28] Fu Y F, Liu J X, Shi X M, et al. Signed cryptographic program verification with typed CryptoLine[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 1591-1606.

[29] Liu J X, Shi X M, Tsai M H, et al. Verifying arithmetic in cryptographic C programs[C]//Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway: IEEE Press, 2020: 552-564.

[30] Hwang V, Liu J X, Seiler G, et al. Verified NTT multiplications for NISTPQC KEM lattice finalists: Kyber, SABER, and NTRU[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022(4): 718-750.

[31] Doychev G, Feld D, Köpf B, et al. CacheAudit: A tool for the static analysis of cache side channels[C]//Proceedings of the 22nd USENIX conference on Security. New York: ACM Press, 2013: 431-446.

[32] Almeida J B, Barbosa M, Barthe G, et al. Verifying constant-time implementations[C]//Proceedings of the USENIX Security Symposium. Berkeley: USENIX Association, 2016: 53-70.

[33] Chen J, Feng Y, Dillig I. Precise detection of side-channel vulnerabilities using quantitative Cartesian Hoare logic[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2017: 875-890.

[34] Antonopoulos T, Gazzillo P, Hicks M, et al. Decomposition instead of self-composition for proving the absence of timing channels[C]//Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM Press, 2017: 362-375.

[35] Yang W K, Vizel Y, Subramanyan P, et al. Lazy self-composition for security verification[C]//Chockler H, Weissenbacher G. Proceedings of the 30th International Conference on Computer Aided Verification. Cham: Springer, 2018: 136-156.

[36] Wu M, Guo S J, Schaumont P, et al. Eliminating timing side-channel leaks using program repair[C]//Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM Press, 2018: 15-26.

[37] Cauligi S, Soeller G, Johannesmeyer B, et al. FaCT: A DSL for timing-sensitive computation[C]//Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM Press, 2019: 174-189.

[38] Watt C, Renner J, Popescu N, et al. CT-wasm: Type-driven secure cryptography for the web ecosystem[J]. Proceedings of the ACM on Programming Languages, 2019, doi: 10.1145/3290390.

[39] Wu M, Wang C. Abstract interpretation under speculative execution[C]//Proceedings of the 40th ACM SIGPLAN

Conference on Programming Language Design and Implementation. New York: ACM Press, 2019: 802-815.

[40] Guo S J, Chen Y Q, Li P, et al. SpecuSym: Speculative symbolic execution for cache timing leak detection[C]// Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. New York: ACM Press, 2020: 1235-1247.

[41] Wang G H, Chattopadhyay S, Biswas A K, et al. KLEESpectre: Detecting information leakage through speculative cache attacks via symbolic execution[J]. ACM Transactions on Software Engineering and Methodology, 2020, 29(3), doi: 10.1145/3385897.

[42] Daniel L A, Bardin S, Rezk T. Binsec/Rel: Efficient relational symbolic execution for constant-time at binary-level[C]//Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2020: 1021-1038.

[43] Guarnieri M, Köpf B, Morales J F, et al. Spectector: Principled detection of speculative information flows[DB/OL]. arXiv preprint:1812.08639, 2018.

[44] Cauligi S, Disselkoen C, Gleissenthall K V, et al. Constant-time foundations for the new spectre era[C]// Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM Press, 2020: 913-926.

[45] Wang G H, Chattopadhyay S, Gotovchits I, et al. oo7: Low-overhead defense against spectre attacks via program analysis[J]. IEEE Transactions on Software Engineering, 2021, 47(11): 2504-2519.

[46] Vassena M, Disselkoen C, Von Gleissenthall K, et al. Automatically eliminating speculative leaks from cryptographic code with blade[J]. Proceedings of the ACM on Programming Languages, 2021, doi: 10.1145/3434330.

[47] Daniel L A, Bardin S, Rezk T. Hunting the haunter—Efficient relational symbolic execution for spectre with haunted RelSE[C]//Proceedings 2021 Network and Distributed System Security Symposium. Reston: Internet Society, 2021, doi: 10.14722/ndss.2021.24286.

[48] Shivakumar B A, Barthe G, Grégoire B, et al. Enforcing fine-grained constant-time policies[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2022: 83-96.

[49] Qin Q, Jiyang J, Song F, et al. DeJITLeak: Eliminating JIT-induced timing side-channel leaks[C]//Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM Press, 2022: 872-884.

[50] Eldib H, Wang C, Schaumont P. Formal verification of software countermeasures against side-channel attacks[J]. ACM Transactions on Software Engineering and Methodology, 2014, 24(2), doi: 10.1145/2685616.

[51] Eldib H, Wang C. Synthesis of masking countermeasures against side channel attacks[C]//Biere A, Bloem R. Proceedings of the 26th International Conference on Computer Aided Verification. Cham: Springer, 2014: 114-130.

[52] Eldib H, Wang C, Taha M, et al. QMS: Evaluating the side-channel resistance of masked software from source code[C]//Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC). Piscataway: IEEE Press, 2014, doi: 10.1145/2593069.2593193.

[53] Barthe G, Belaïd S, Dupressoir F, et al. Verified proofs of higher-order masking[C]//Oswald E, Fischlin M. Proceedings of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology—EUROCRYPT 2015. Berlin, Heidelberg: Springer, 2015: 457-485.

[54] Barthe G, Belaïd S, Dupressoir F, et al. Strong non-interference and type-directed higher-order masking[C]// Proceedings of the 2016 ACM SIGSAC Conference on

Computer and Communications Security. New York: ACM Press, 2016: 116-129.

[55] Blot A, Yamamoto M, Terauchi T. Compositional synthesis of leakage resilient programs[C]//Maffei M, Ryan M. Proceedings of the 6th International Conference on Principles of Security and Trust. Berlin, Heidelberg: Springer, 2017: 277-297.

[56] Coron J S. Formal verification of side-channel countermeasures via elementary circuit transformations[C]// Preneel B, Vercauteren F. Proceedings of the 16th International Conference on Applied Cryptography and Network Security. Cham: Springer, 2018: 65-82.

[57] Bloem R, Gross H, Iusupov R, et al. Formal verification of masked hardware implementations in the presence of glitches[C]//Nielsen J B, Rijmen V. Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology—EUROCRYPT 2018. Cham: Springer, 2018: 321-353.

[58] Zhang J, Gao P F, Song F, et al. SCInfer: Refinement-based verification of software countermeasures against side-channel attacks[C]//Chockler H, Weissenbacher G. Proceedings of the 30th International Conference on Computer Aided Verification. Cham: Springer, 2018: 157-177.

[59] Gao P F, Zhang J, Song F, et al. Verifying and quantifying side-channel resistance of masked software implementations[J]. ACM Transactions on Software Engineering and Methodology, 2019, 28(3), doi: 10.1145/3330392.

[60] Wang J B, Sung C, Wang C. Mitigating power side channels during compilation[C]//Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM Press, 2019: 590-601.

[61] Barthe G, Belaïd S, Cassiers G, et al. MaskVerif: Automated verification of higher-order masking in presence of physical defaults[C]//Sako K, Schneider S, Ryan P Y A. Proceedings of the 24th European Symposium on Research in Computer Security—ESORICS 2019. Cham: Springer, 2019: 300-318.

[62] Belaïd S, Goudarzi D, Rivain M. Tight private circuits: Achieving probing security with the least refreshing[C]// Peyrin T, Galbraith S D. Proceedings of the 24th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology—ASIACRYPT 2018. Cham: Springer, 2018: 343-372.

[63] Belaïd S, Dagand P É, Mercadier D, et al. Tornado: Automatic generation of probing-secure masked bitsliced implementations[C]//Canteaut A, Ishai Y. Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology—EUROCRYPT 2020. Cham: Springer, 2020: 311-341.

[64] Knichel D, Sasdrich P, Moradi A. SILVER—Statistical independence and leakage verification[C]//Moriai S, Wang H. Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology—ASIACRYPT 2020. Cham: Springer, 2020: 787-816.

[65] Belaïd S, Coron J S, Prouff E, et al. Random probing security: Verification, composition, expansion and new constructions[C]//Micciancio D, Ristenpart T. Proceedings of the 40th Annual International Cryptology Conference, Advances in Cryptology—CRYPTO 2020. Cham: Springer, 2020: 339-368.

[66] Gao P F, Xie H Y, Song F, et al. A hybrid approach to formal verification of higher-order masked arithmetic programs[J]. ACM Transactions on Software Engineering and Methodology, 2021, 30(3), doi: 10.1145/3428015.

[67] Cassiers G, Grégoire B, Levi I, et al. Hardware private circuits: from trivial composition to full verification[J]. IEEE Transactions on Computers, 2021, 70(10): 1677-1690.

[68] Gao P F, Xie H Y, Sun P, et al. Formal verification of masking countermeasures for arithmetic programs[J]. IEEE Transactions on Software Engineering, 2022, 48 (3): 973-1000.

[69] Belaïd S, Mercadier D, Rivain M, et al. IronMask: Versatile verification of masking security[C]//Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2022: 142-160.

Status and Prospects of Formal Verification for Security of Cryptographic Implementations

SONG Fu

School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China

Abstract Serving as a foundation, cryptography is the key technique for ensuring network and information security. However, cryptographic implementations suffer from various security threats. Formal verification is the most important technique for rigorously proving the security of cryptographic implementations. This article first reviews the development status and application of formal verification techniques for cryptographic implementations both in China and abroad in terms of functional correctness, memory security, and side-channel security including time and power consumption. Then the article summarizes the drawbacks, current challenges, and development trends of these formal verification techniques for cryptographic implementations. According to the security requirements and limited progress in formal verification techniques for cryptographic implementations in China, it is suggested that high-performance and high-assurance cryptographic libraries should be constructed. In addition, the article proposes to follow the top-down design principle, strengthen independent innovation, and unify intermediate representation language under the guidance of national key task, and it plans to develop a high-efficiency, high-precision, and full-automated formal verification platform and a secure compiler optimization toolchain, create high-performance and high-assurance common cryptographic library with minimized trusted computing base, and promote the integrated development of both academic and industrial communities, so as to improve national network and information security.

Keywords cryptographic implementations; security and privacy; formal verification; functional correctness; memory security; side-channel security

宋富，上海科技大学信息科学与技术学院长聘副教授，研究员。上海科技大学系统与安全中心主任。研究方向为形式化验证、系统安全和人工智能安全。主持和参与多项国家自然科学基金青年、面上、重点和中德国际合作项目。发表论文70余篇。获欧洲软件科学与技术协会最佳论文奖、亚马逊研究奖和上海市“浦江人才”、上海市“晨光学者”荣誉称号。

电子信箱：songfu@shanghaitech.edu.cn。