様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　５　月　２６　日現在

機関番号：　１３９０４
研究種目：　若手研究(B)
研究期間：　2014～2016
課題番号：　２６８７０２７８

研究課題名（和文）秘匿化・難読化のためのPLCプログラムのハードウェア化に関する研究

研究課題名（英文）Research on Hardware Implementation Methodology of PLC Programs for Concealment and Obfuscation

研究代表者

藤枝　直輝（FUJIEDA, Naoki）

豊橋技術科学大学・工学（系）研究科（研究院）・助教

研究者番号：３０７０８４２５

交付決定額（研究期間全体）：（直接経費）　　1,800,000円

研究成果の概要（和文）：本課題では，シーケンス制御に用いられる制御用計算機であるPLC向けのプログラムを，FPGAを用いたハードウェアに変換することで，解析の困難さを高める手法について研究した．難読化手法を適用した上でハードウェア化することを検討したほか，そのためのソフトウェアインフラの構築を行った．また，プログラムの一部のみをハードウェア化し，残りをソフトウェアで実行することを見越して，汎用プロセッサ上のソフトウェア難読化・暗号化の技術についても研究した．

研究成果の概要（英文）：This theme examined methods to conceal and obfuscate programs for PLCs, computers designed for sequence control, by translating them into hard-wired circuits using FPGAs. Translation after applying an obfuscation method was studied and software infrastructure to obtain obfuscated circuits was developed. This theme also studied techniques for obfuscating and encrypting software on general purpose processors, in expectation of the case that only a part of a program was implemented by hardware and the rest was executed on a processor as software.

研究分野：　計算機システム

キーワード：　FPGA　ハードウェア　セキュリティ　専用回路　難読化　セキュアプロセッサ

１．研究開始当初の背景

PLC (Programmable Logic Controller) は，産業用機械等のシーケンス制御において広く用いられる制御用計算機である．近年は大規模な制御においても PLC が利用されており，公共インフラを支える技術のひとつとなっている．PLC に関しては，Stuxnet とよばれるマルウェアが2010年に発見されて以降，そのセキュリティに関する関心が高まっている．このマルウェアは特定の PLC を監視する PC に感染し，PLC の制御を改ざんすることが目的であった．このような脅威から PLC プログラムを保護することは，ノウハウの保護や産業用システムの安全のために重要である．

PLC プログラムを保護する手法の候補の1つは，PLC プログラムを FPGA（Field-Programmable Gate Array）による論理回路，ハードウェアに変換するアプローチである．ハードウェア化の主な効能は，高速実行により応答性が改善することと，動作の等価性を保ちつつも内部の解析が困難になることである．前者については本課題の研究開始前からも多くの研究があった．しかし後者の，いわゆる難読化の効能についてはそれまであまり検討されていなかった．そのため，ハードウェア化の設計の変化がプログラムの秘匿性に与える影響，そして，PLC プログラムを効果的に保護する設計方法については，本課題の研究開始の時点において定かではなかった．

２．研究の目的

本課題の研究開始の時点においては，PLC プログラムのハードウェア化にあたり，以下のことを明らかにすることを目的とした．

(1) より解析や盗用が困難なハードウェア記述を生成するための手法を提案すること．
(2) そのためのソフトウェアインフラストラクチャを構築することで，これらを容易に適用できるようにすること．
(3) 生成後のハードウェア記述や回路の評価を通じ，単に既存のハードウェア化およびその難読化手法を適用したものと比べ，提案手法がより効果的な難読化を達成することを定量的に明らかにすること．

これにより，ハードウェア化された PLC プログラムの解析困難性の効能を明らかにし，ノウハウの保護や産業用システムの安全に資することをめざした．

３．研究の方法

本課題の計画段階においては，PLC プログラムを直接ハードウェア記述に変更することを前提に，ソフトウェアインフラストラクチャの整備，および難読化手法の検討・評価を段階的に行うとしてきた．しかし，近年の高位合成技術（C 言語等で記述されたソフトウェアからハードウェア記述を得る技術）の高まりから，PLC プログラムを一度 C 言語などのプログラムに変換し，それを既存の高位合成ツールと組み合わせてハードウェア記述を得る，という手法を新たに検討することとした．

この手法の利点として，プログラムの中で特に秘匿化が必要である部分のみをハードウェア化し，残りの部分は汎用プロセッサで実行するといった，ハードウェアとソフトウェアとの協調が容易であることが挙げられる．またこの場合，汎用プロセッサで実行する部分にもある程度の秘匿化が求められると考えるのが妥当である．そのため，汎用プロセッサ上のソフトウェア難読化・暗号化の要素技術について併せて研究を行うことが有効である．

以上のことから，本研究の内容は大きく分けて，(1) 難読化・ハードウェア化のためのインフラ構築，(2) ハードウェア化における難読化手法の適用，および (3) 汎用プロセッサ上のソフトウェア難読化・暗号化の要素技術，に分類される．

(1) 難読化・ハードウェア化のインフラ

PLC プログラムを直接ハードウェア記述に変換する手法のソフトウェアインフラは，プログラムを解析し中間形式へと変換するフロントエンド部と，中間形式同士で変換を行う中間処理部，中間形式をハードウェア記述へ変換するバックエンド部からなるツールチェインとして構築した．中間形式として XML を採用し，PLC プログラムの命令と 1 対 1 対応する命令レベルの中間形式と，PLC プログラムの表現として広く用いられるラダー図の段と 1 対 1 対応する動作レベルの中間形式を用意した．これには，ハードウェア化を単一のツールとして実現する場合 ① と比べ，個々のツールの複雑度を下げ，プログラムの見通しの良さと保守性を高めるねらいがある．

一方，高位合成を利用する手法においては，過去に本研究室で開発された，PLC プログラムを C 言語に変換するツールを使用した．ここでは，命令と 1 対 1 対応された C 言語記述を順序通りに生成し，プログラム全体は 1 つの関数として実現される．この関数を Xilinx 社の高位合成ツールである Vivado HLS に与えることで，ハードウェア記述を得る．

いずれの手法においても，FPGA と ARM アーキテクチャのプロセッサを同一チップに集積した，Xilinx 社の Zynq-7000 シリーズを動作環境としてサポートした．Zynq 上に実現される，ハードウェア化された PLC プログラムを含むシステムのブロック図を図 1 に示す．ハードウェア化された PLC プログラムは，FPGA 部分（Programmable Logic）上のコプロセッサ (PLC System) として搭載される．すなわち，プロセッサから起動指令とデータを受取り，プログラムに相当する処理を行い，プロセッサにデータを返却する，という一連の処理を行う．PLC プログラムの

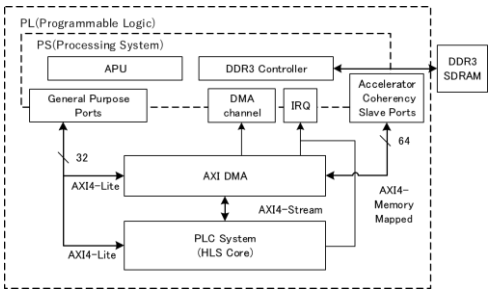


図 1 ハードウェア化された PLC プログラムを含むシステムのブロック図.

うちハードウェア化を行わない部分や，その他の汎用的な処理は，プロセッサによりソフトウェアで処理される.

(2) 難読化手法の適用

難読化手法として，ソフトウェアの難読化手法として提案されている Opaque Predicate [②] について検討を行った．これは，プログラム中に必ず成立または必ず不成立となる分岐を加えることで，プログラムの制御の流れを複雑化するものである．当然ながら，追加された分岐が必ず成立（不成立）するものであると容易に看破できてしまっては，この難読化は意味をなさない．そのため，これを防ぐための有向グラフを利用した方法が併せて提案されている.

PLC プログラムを直接ハードウェア化する場合, 中間処理部のツールを 1 つ追加して, 中間形式に難読化に関するマークアップを追加することでこれを実現した．上述の有向グラフは対応するハードウェア記述を直接生成して実装した.

一方，高位合成ツールを利用する場合では, C 言語のプログラムを難読化された C 言語のプログラムに変換するツールを作成し，これを高位合成の前に適用することでこれを実現した. 有向グラフは対応する C 言語を生成し，プログラム本体とともに高位合成することによりハードウェア実装した.

(3) 汎用プロセッサ上の難読化・暗号化技術

本研究では，① 命令レジスタファイルとよばれるプロセッサ技術をソフトウェア難読化に用いるための手法の改良と，② パス型 ORAM とよばれるメモリアクセス秘匿化技術の改良について提案した．以下，これらのそれぞれについて述べる.

① 命令レジスタファイルによる難読化

命令レジスタファイル（IRF）[③] は，頻繁に使われるプロセッサの命令を記憶した表のことで，電話で例えるところの「短縮ダイアル」に相当するものである．IRF に格納された命令は，あらかじめ「IRF の○番」といったより短い表現に変換され，実行時に IRF を参照して元の機械語表現に復元される．その中身が外部から隠蔽されていれば，短い表現から元の命令を推測することは難しく，難読化を実現できる.

本研究では，IRF をソフトウェアの難読化に応用するために必要な技術や，その機能を補うための追加の機構，あるいは IRF 自体の効率化について検討・実装・評価を行った.

② パス型 ORAM の改良

ソフトウェアの保護においては，プログラムやデータを暗号化するだけでは不十分であり，どの順番にデータにアクセスしているか，すなわちアクセスパターンを解析することでも情報が流出する場合があることが指摘されている．これを解決し，アクセスパターンを外部から隠蔽する手法のことを忘却型 RAM（Oblivious RAM; ORAM）とよぶ. この中でも特に軽量に実現できるものとして注目されているのが，パス型 ORAM [④] である.

本研究では，パス型 ORAM により生成される隠蔽されたアクセスパターンが冗長なアクセスを含むことに注目し，これを取り除く方法について検討を行った.

4．研究成果

(1) 難読化・ハードウェア化のインフラ

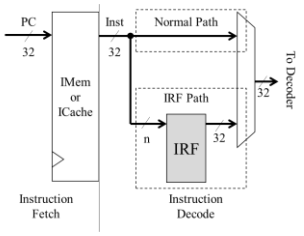
前述の通り, PLC 命令列から直接ハードウェア記述を得る方法と，高位合成を利用する方法とで，同一の動作環境が利用できるようにツールを構築した．これにより，両者の方法をこの環境上で定量的に比較することが可能となった.

実際にいくつかの PLC プログラムを対象に生成される回路を比較したところ，両者の方法で回路規模や動作周波数に大きな差は見られないことが確認できたが，高位合成を利用するケースでは回路規模と動作周波数との間のトレードオフを自動的に考慮している様子も確認できた．これらの知見をまとめた論文は査読付国際会議 ISIE2017 に採択され，近日発表予定である.

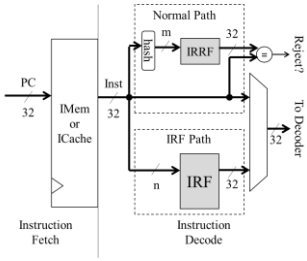
(2) 難読化手法の適用

Opaque Predicate の実装においては，分岐が必ず成立する（しない）ことが合成ツールによって看破されれば，その部分が最適化により除去されてしまう．それぞれの方法で Opaque Predicate を実装し, いくつかの PLC プログラムに対して実際に難読化された回路を生成し評価したところ，いずれの方法でも動作の同一性を保ちながら回路規模が増大した．つまり，少なくとも合成ツールによっては看破されない Opaque Predicate がハードウェア上に実現できたと考えられる.

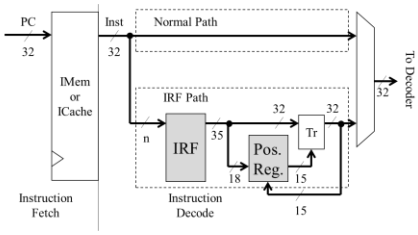
こうした成果については国内研究会にて発表を行った．また，今後具体的な比較・評価を行うためのいくつかの基礎データが得られたが，実際の比較・評価は今後の課題である.



IRF 単体



IRF + 命令拒否レジスタファイル(IRRF)



IRF + 位置レジスタ（Pos. Reg.）

図 2　IRF とその機能を補う機構.

(3) 汎用プロセッサ上の難読化・暗号化技術

① 命令レジスタファイルによる難読化

IRF をソフトウェアの難読化に応用するにあたっては，本課題の研究開始の時点において，ある程度大きなサイズ（1024 命令程度）の IRF を用い，格納する命令の選択を工夫することで，より効果的な保護が可能であることを明らかにしていた．本研究では，命令の選択アルゴリズムを改良することで，1024 命令の場合で従来と比べ約 400〜35,000 倍高速に準最適な命令の組合せを得られるようになった．こうした成果は英文論文として論文誌 Microprocessors and Microsystems に採録された.

また, IRF の機能を補うための機構として，命令拒否レジスタファイル（IRRF）を提案するとともに，IRF とともに提案されていた位置レジスタ[③] を難読化に活用するための方法を検討した．これらの機構のハードウェア実装における位置付けを図 2 に示す．IRRF は IRF とは並列に，位置レジスタは IRF による命令変換の後処理として実現される．

IRRF は，IRF のみでは実現できなかったソフトウェアの改ざん防止の機能を提供する．IRF に格納された命令が，命令メモリ上であらかじめ短い表現にすべて変換されていれば，その命令の元の表現は命令メモリに存在しえない．もし現れれば，それは改ざんによって不正に生成されたものだと判断できる．これを検出するのが IRRF である．これにかかる成果は査読付国際会議 PPREW-5 で発表した.

位置レジスタは，複数の相異なる命令を 1 つの抽象化された命令表現に統合することで，より多くの命令を IRF から実行させることを目的としている．ここでは，命令が使用するレジスタ番号を，「直前の命令の書き込み先レジスタ」のように過去の命令が使用したレジスタ番号を用いて抽象化する．これらを記憶し，抽象化された命令表現から元の命令を復元するのが位置レジスタの役割である．これにかかる成果は国内研究会での発表を経ており，英文論文として論文誌への投稿を予定している.

さらに，IRF 自体の効率化についても研究を行った．上述の抽象化には，過去の命令が使用したレジスタ番号を使う方法の他に，命令に付加されたパラメータを使う方法も提案されている．本研究ではパラメータの使い方を改良し，より多くの命令を IRF から実行できるようにした．かかる成果は英文論文として電気学会共通英文論文誌に採録された．当該論文では IRF の当初の用途である消費電力削減への有効性を示したが，改良自体は本研究で注目する難読化にも活用できる可能性がある．その活用は今後の課題である.

② パス型 ORAM の改良

パス型 ORAM により生成される隠蔽されたアクセスパターンの冗長性を省くために，Last Path Caching という機構を提案した．この冗長性は一度書き出したデータをすぐに読み出してしまうことに起因している．そのため，これを擬似的にキャッシュする機構を追加することで冗長性が取り除けることを明らかにし，評価を通じて類似の手法と比べた優位性を明らかにした．この成果は査読付国際学会 CSA-4 で発表したほか, 英文論文として論文誌 International Journal of Networking and Computing に採録された.

＜引用文献＞

① S. Ichikawa et al.: An FPGA implementation of hard-wired sequence control system based on PLC software, IEEJ Transactions on Electrical and Electronic Engineering, 6(4)367–375, 2011.
② C. Collberg et al.: A taxonomy of obfuscating transformations, Technical report 148, Department of Computer Science, The University of Auckland, New Zealand, 1997.
③ S. Hines et al: Improving program efficiency by packing instructions into registers, 32nd Annual International Symposium on Computer Architecture (ISCA2005), 260–271, 2005.
④ E. Stefanov et al: Path ORAM: An Extremely Simple Oblivious RAM Protocol, 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS '13), 299–310, 2013.

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計　３件）

① Naoki Fujieda, Ryo Yamauchi, Hiroki Fujita, and Shuichi Ichikawa: A Virtual Cache for Overlapped Memory Accesses of Path ORAM, International Journal of Networking and Computing（掲載決定）, 2017.査読有.
② Naoki Fujieda, Tasuku Tanaka, and Shuichi Ichikawa: Design and Implementation of Instruction Indirection for Embedded Software Obfuscation, Microprocessors and Microsystems, 45(A) 115–128, 2016. 査読有.
DOI: 10.1016/j.micpro.2016.04.005
③ Naoki Fujieda and Shuichi Ichikawa, An XOR-based Parameterization for Instruction Register Files, IEEJ Transactions on Electrical and Electronic Engineering, 10(5)592–602, 2015. 査読有.
DOI:10.1002/tee.22123

〔学会発表〕（計１２件）

① Naoki Fujieda, Shuichi Ichikawa, Yoshiki Ishigaki, and Tasuku Tanaka: Evaluation of the hardwired sequence control system generated by high-level synthesis, 2017 IEEE International Symposium on Industrial Electronics (ISIE 2017), June 19–21, 2017 (accepted), Edinburgh, Scotland (UK).
② Naoki Fujieda, Ryo Yamauchi, and Shuichi Ichikawa: Last Path Caching: A Simple Way to Remove Redundant Memory Accesses of Path ORAM, 4th Workshop on Computer Systems and Architectures (CSA-4) held in conjunction with CANDAR '16, November 22–25, 2016, 東広島芸術文化ホール（広島県東広島市）.
③ Naoki Fujieda, Kiyohiro Sato, and Shuichi Ichikawa: A complement to Enhanced Instruction Register File against Embedded Software Falsification, 5th Program Protection and Reverse Engineering Workshop (PPREW-5), December 8, 2015, Los Angeles, CA (USA).

〔その他〕
論文プレプリント等は，研究代表者の個人Webページに掲載している．
https://sites.google.com/site/nfproc

６．研究組織
(1)研究代表者
藤枝　直輝（FUJIEDA, Naoki）
豊橋技術科学大学・工学（系）研究科（研究院）・助教
研究者番号： 30708425