

ソフトウェアに対する配信時電子透かし 埋め込みシステムの提案と実装

19622026 永山 涼雅 情報セキュリティ研究室

背景

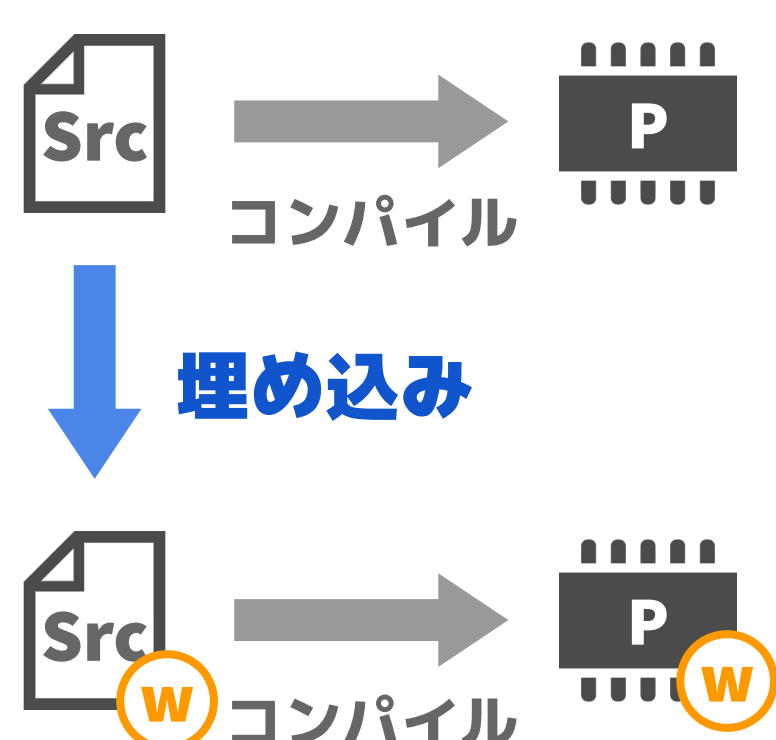
ソフトウェアの違法コピー・違法アップロードの増加
→ ソフトウェアの**権利保護技術**の必要性

電子透かし

- ・ 微小な変化を与えることで情報を埋め込む技術
- ・ 利用者の情報を埋め込むことで違法行為を**抑止**できる

既存研究

ソースコードに透かしを埋め込む^{[1][2]}



- ・ 埋め込み方法が**プログラミング言語に依存**
- ・ 埋め込み情報ごとに**再コンパイルが必要**

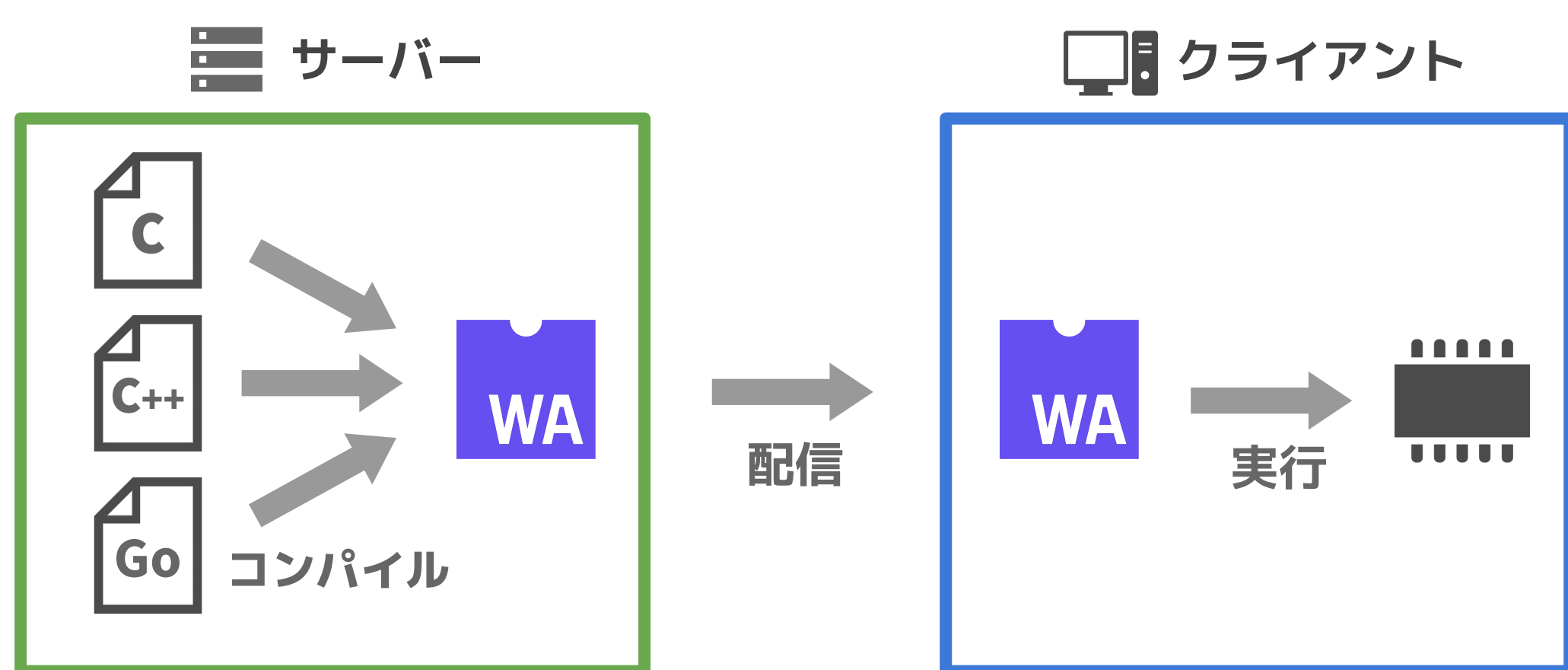
目的 実用性を損なわないソフトウェア透かしシステムの構築

[1]: A. Monden, H. Iida, K. ichi Matsumoto, K. Inoue, and K. Torii, "Watermarking java programs," ISFST99, October 1999
[2]: A. Fionov, "Digital Watermarks for C/C++ Programs," 7TH FRUCT, 2009

WebAssembly

WebAssembly^[3]

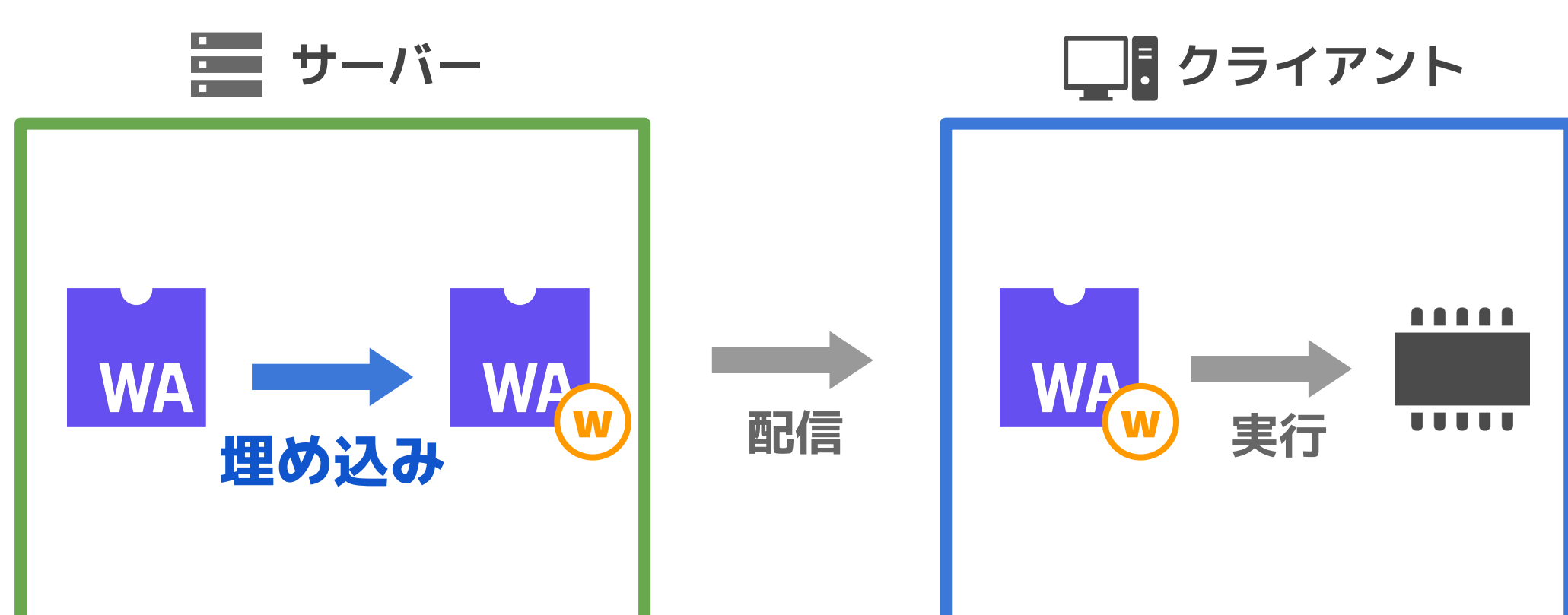
- ・ スタックマシン型の仮想マシン上で動作する低級言語
- ・ Webブラウザ上などでバイトコードを実行できる
- ・ JavaScriptより高速に解析・実行される
- ・ 様々な言語からWebAssemblyへコンパイルできる



[3] webassembly.org, "WebAssembly," <https://webassembly.org>

提案システム

- ・ **WebAssembly**に透かしを挿入する
- ・ **配信時**にユーザ情報などを埋め込む

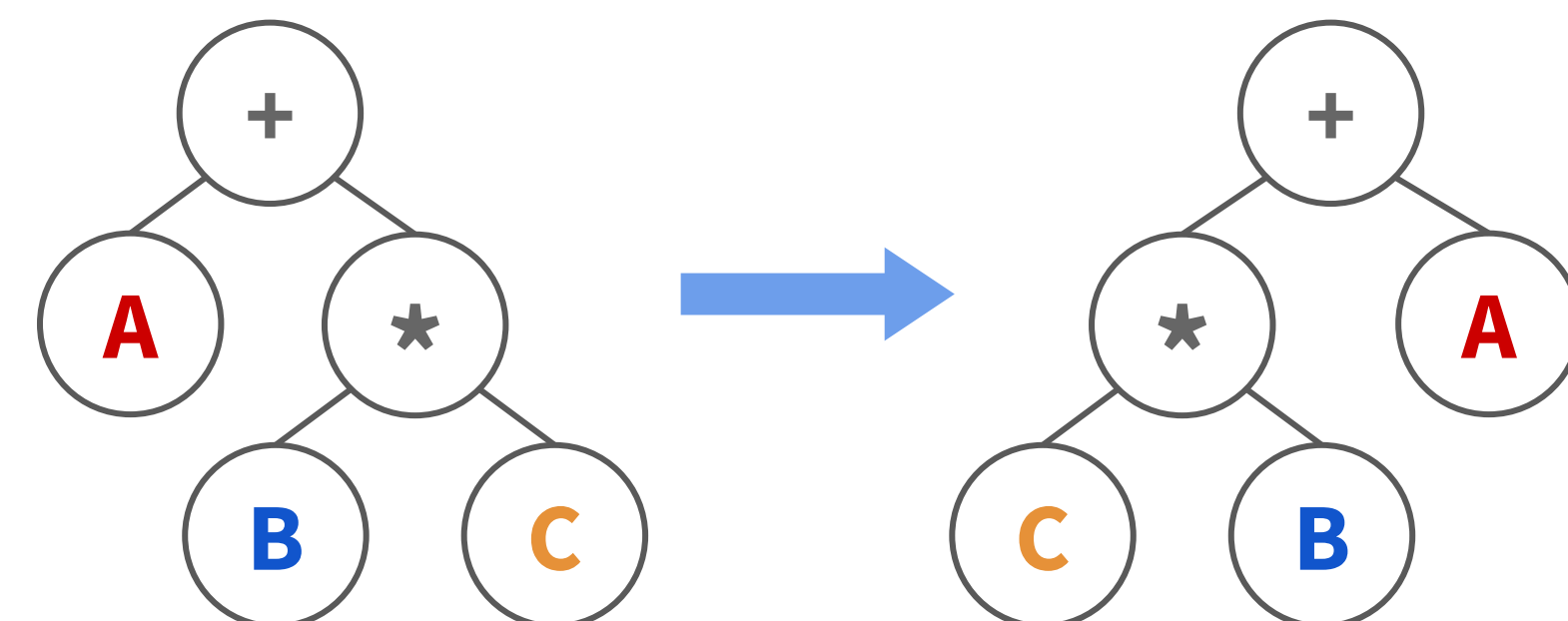


埋め込み手法

埋め込み手法1：命令オペランドの入れ替え

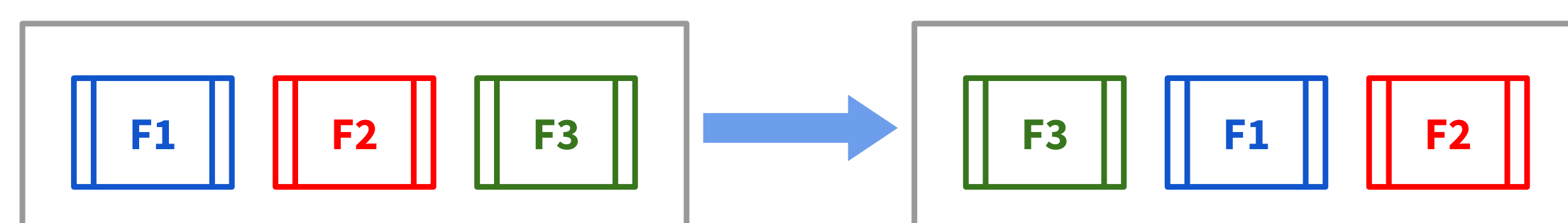
- ・ **可換な命令**のオペランドを入れ替える

例: $A + B * C$



埋め込み手法2：関数の順序を変更

- ・ プログラム内の関数の定義順を変更する



評価

埋め込み量

	バイナリサイズ [B]	関数の数	手法1 [bit]	手法2 [bit]
styled-jsx/ mappings.wasm	48,693	45	3,130	90
zlib.wasm	61,863	58	3,170	120
ammo.js/ ammo.wasm	687,520	1,640	37,227	3,444

実行速度

- ・ **zlib^[4]** deflateベンチマークの実行時間 (N = 50)



- ・ **ammo.js^[5]** ベンチマークの実行時間 (N = 50)



[4]: G. Roelofs, M. Adler, J. Gailly, "zlib," <https://www.zlib.net/>
[5]: Alon Zakai, "ammo.js," <https://github.com/kripen/ammo.js>

まとめ

利点

- ・ プログラムの実行速度を損なわない
- ・ プログラムの開発言語に依存しない
- ・ 配信時に情報を埋め込むため、再コンパイルが不要

課題

- ・ 上書き攻撃への耐性