Modular Verification of Probabilistic Constant-Time

Xingyu Xie

Max-Planck Institute for Security and Privacy

Jan 11, 2026



MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

1. Background: (probabilistic) constant-time
2. Research Question: modular verification for (probabilistic) constant-time
3. Proposed Solution: inspired by sequential modular composition of multiparty computation
4. Conclusions and Future Works

1 Background

2 Research Question

3 Proposed Solution

4 Conclusions and Future Works

Constant-time

Constant-time: the adversary cannot infer secret information by observing branch decisions and memory accesses, i.e., the leakages have no secret information.

```
1 leak b;
2 if (b) {
3    leak x;
4    y = a[x];
5 }
```

1 Background

2 Research Question

3 Proposed Solution

4 Conclusions and Future Works

Modular Verification of Constant-Time

Why modularity? structural, easy to understand and maintain, reusability.

How to prove constant-time in a *modular* way along function calls?

```
1 int reject_sampling(int bound) {
2     ...
3 }
4
5 int otp(int x) {
6   ...
7   r = reject_sampling(2048);
8   ...
9 }
```

Modularized constant-time (MCT)

contract: labelling public/secret for input/output, "public" means being fine to be observed by adversary.

modularized constant-time w.r.t. contract: the leakages have no more secret information than public input and public output.

How to use the established contracts of callees when proving a caller?

```
1  // input:  ...(secret) ...(public)
2  // output:  ...(secret) ...(public)
3  int reject_sampling(int bound) {
4    ...
5  }
6
7  int caller() {
8    ...
9    r = reject_sampling(2048);
10   ...
11 }
```

1 Background

2 Research Question

3 Proposed Solution

4 Conclusions and Future Works

Modular Composition of CT

Theorem

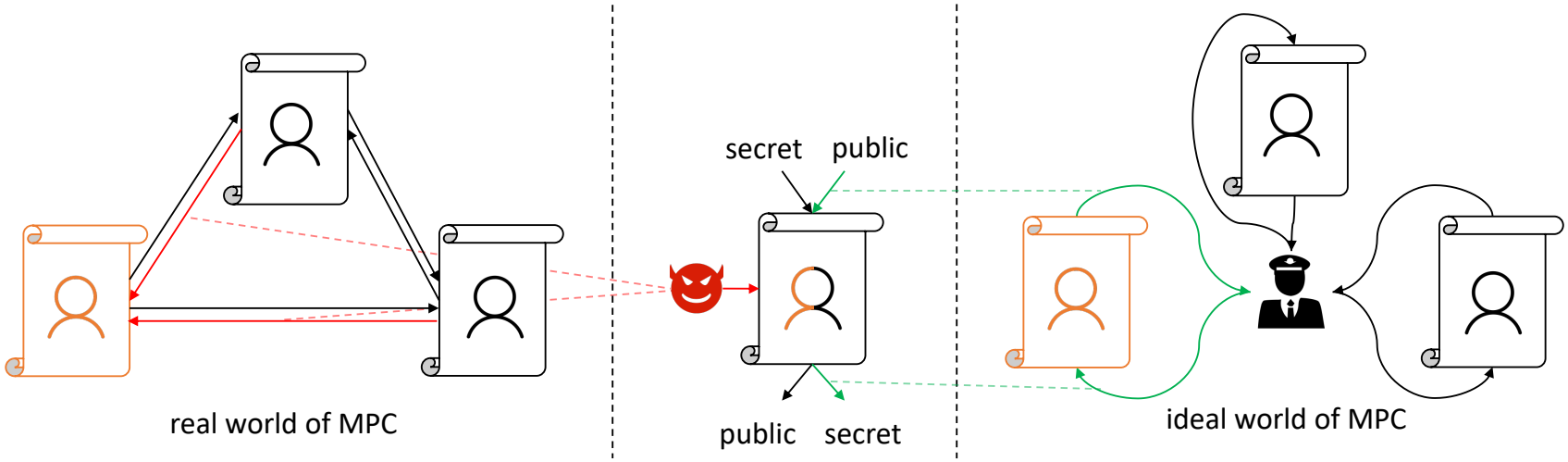
Let $(\!| P |\!)$ *be the program replacing all function calls* $res = Q(arg)$ *in* P *by*

$$leak\ arg_{pub}; res = Q^{ideal}(arg); leak\ res_{pub},$$

where Q^{ideal} *is functionally equivalent to* Q *but has no leakage. If all callees are MCT and the replaced caller* $(\!| P |\!)$ *is MCT, then the original caller* P *is MCT.*

```
1  // in:  bound (public)
2  // out:  r (public)
3  int reject_sampling
4            (int bound) {
5    ...
6  }
7
8  int caller(int x) {
9    ...
10   leak 2048;
11   int r = $ [0..2047];
12   leak r;
13   ...
14 }
```

Sequential Modular Composition of Multiparty Computation



Security definition: red ⊆ green

Theorem (4.5, Ran Canetti, JoC'2000)

Consider a protocol π which is able to ideally compute function f. If protocol ρ securely computes f, and π is secure, then the composed protocol π^ρ is secure.

1 Background

2 Research Question

3 Proposed Solution

4 Conclusions and Future Works

Conclusions and Future Works

Summary: a call-by-contract-style modular verification approach for (probabilistic) constant-time.

Future works

- practice: relational Hoare logic
- generalization: non-interference (e.g., obliviousness)