Software Protection through Dynamic Code Mutation

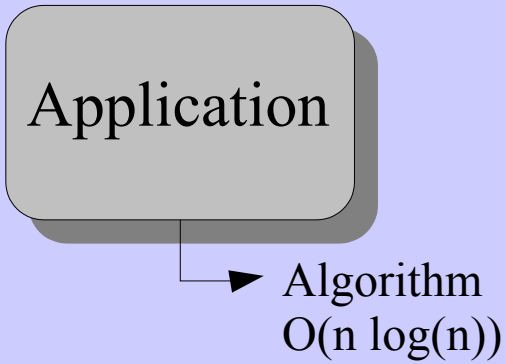
Matias Madou*, Bertrand Anckaert*, Patrick Moseley#, Saumya Debray#, Bjorn De Sutter* and Koen De Bosschere*

* Ghent University
#University of Arizona

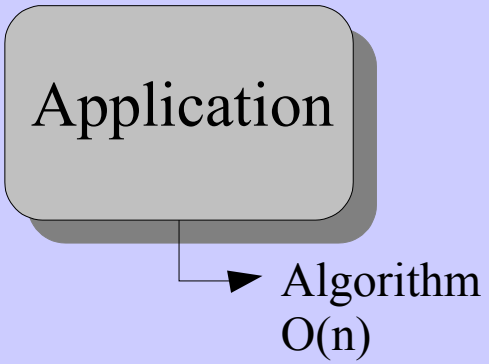
Conference: WISA05; International Workshop on Information Security Applications

Problem

Software company A

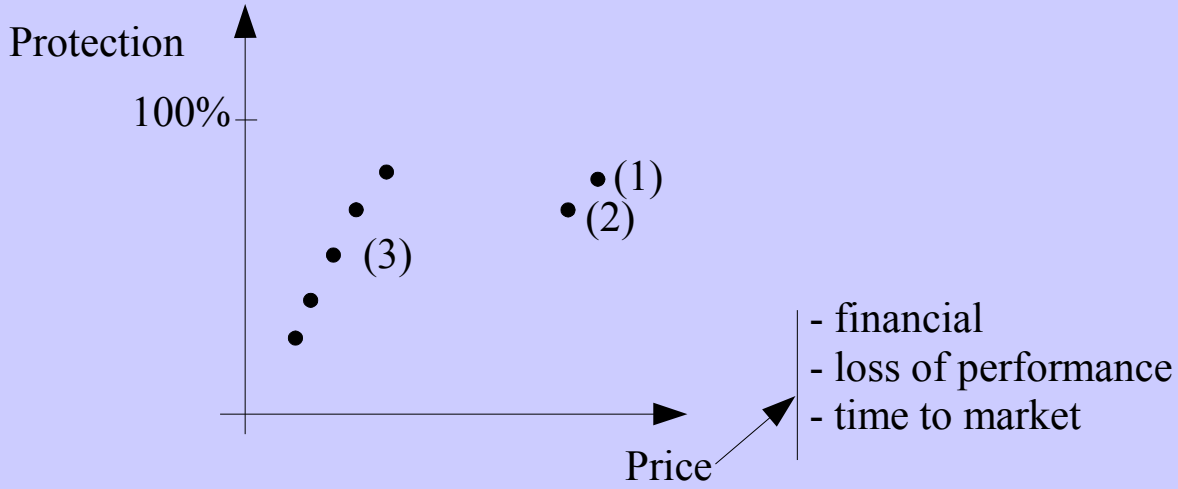


Software company B



How can company B protect their new algorithm?

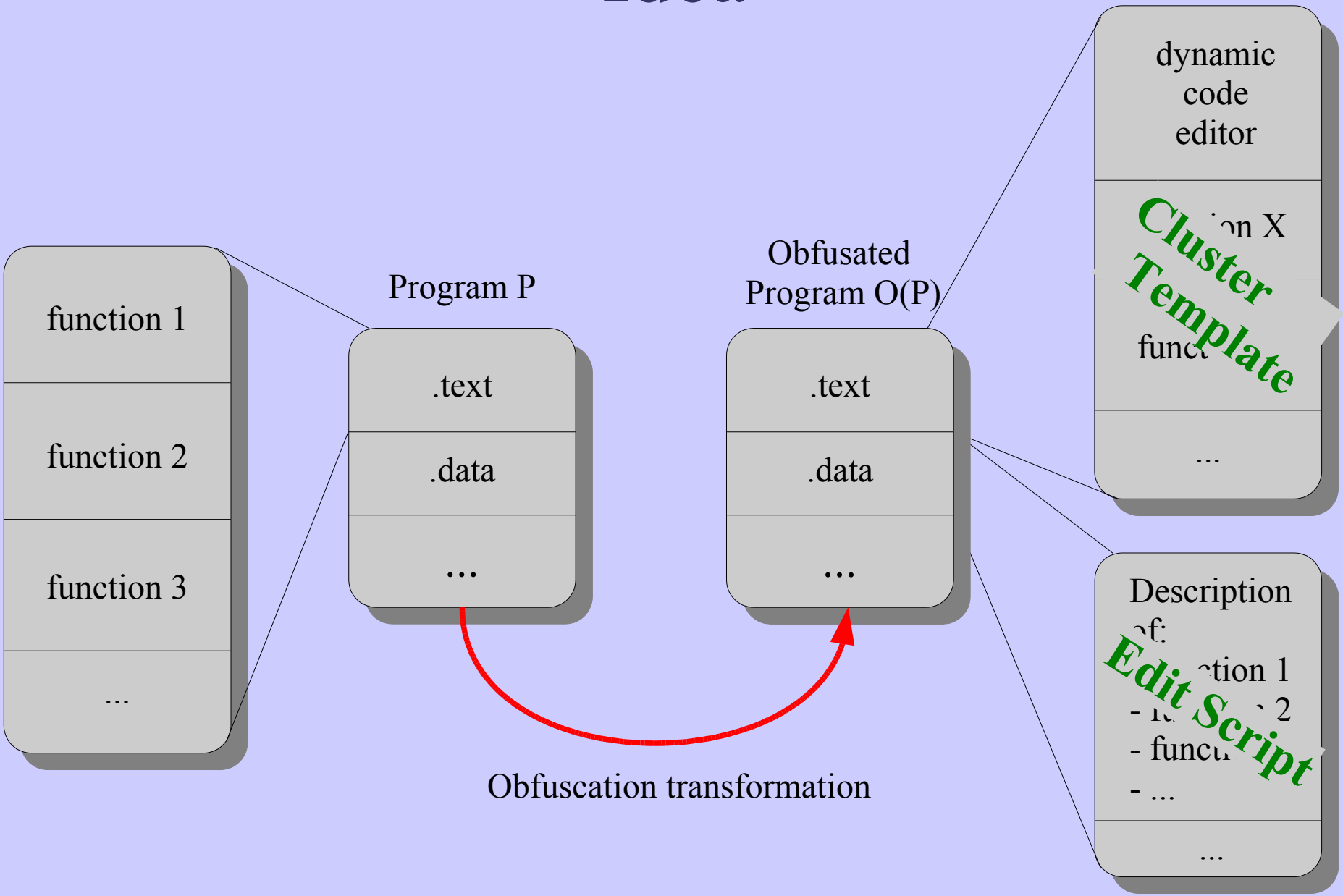
1) to patent
2) special hardware
3) obfuscation



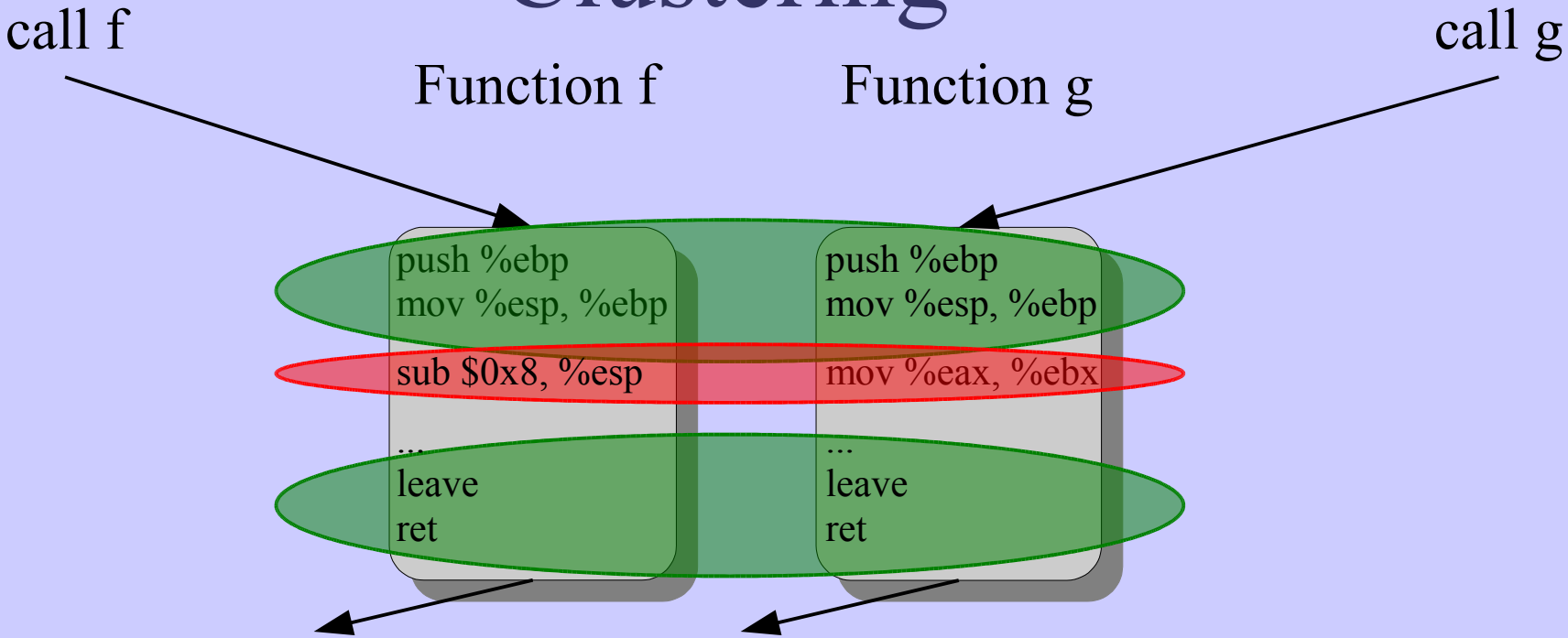
Obfuscation

- Goal: Hide the internal working of a program
- How: Program transformations to complicate program analysis
- In this case: Mutate a program as it executes
  Goal: region of memory occupied by many different code sequences during execution

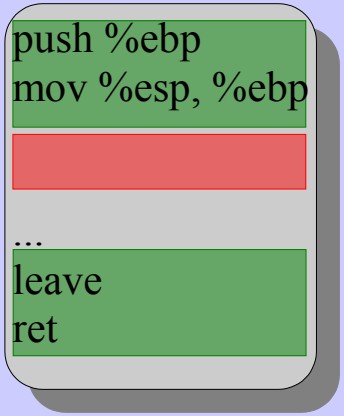
Idea


function 1
function 2
function 3
...
Program P
.text
.data
...
Obfuscation transformation
Obfusated
Program O(P)
.text
.data
...
dynamic
code
editor
Cluster
Template
...
Description
Edit Script
...

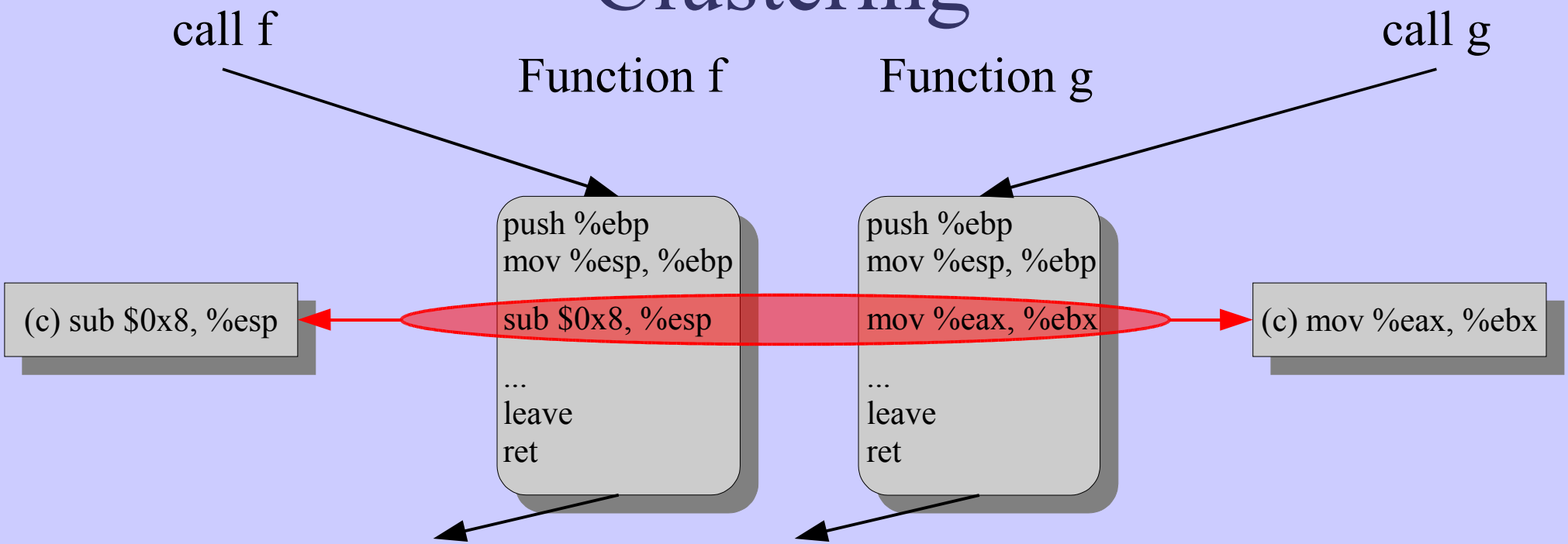
Clustering

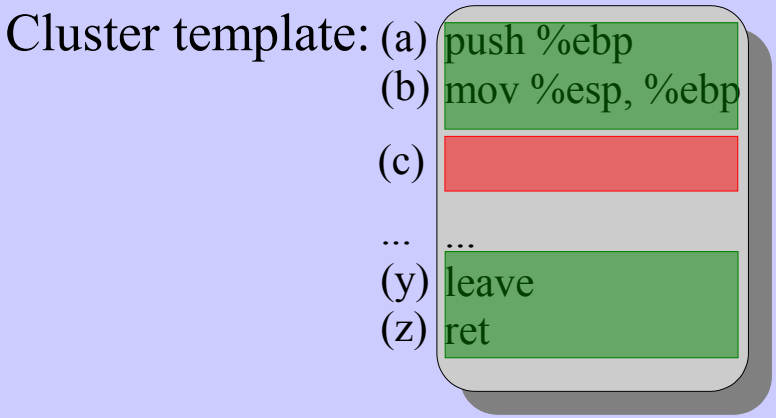


Function f or g

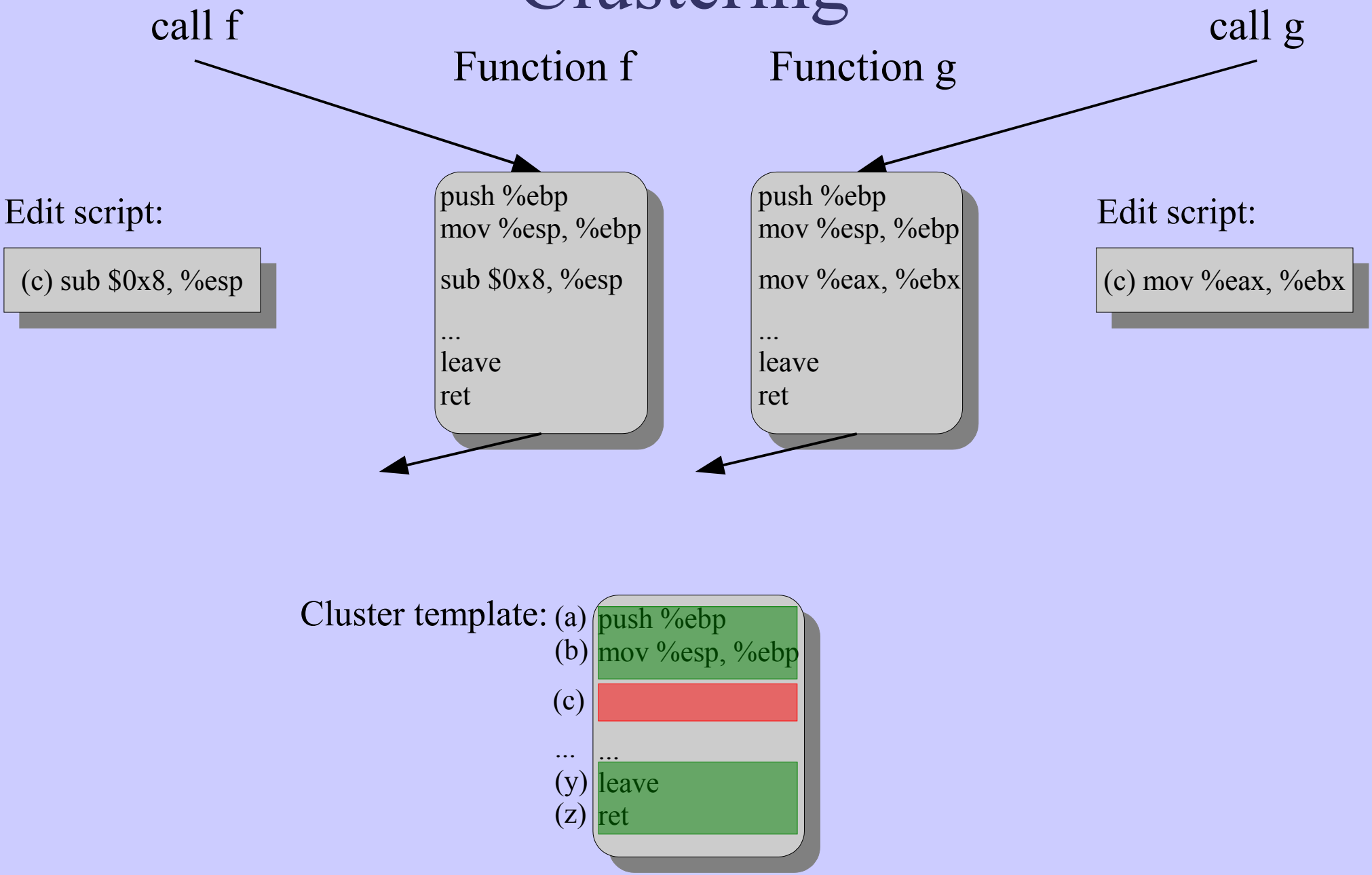


Clustering

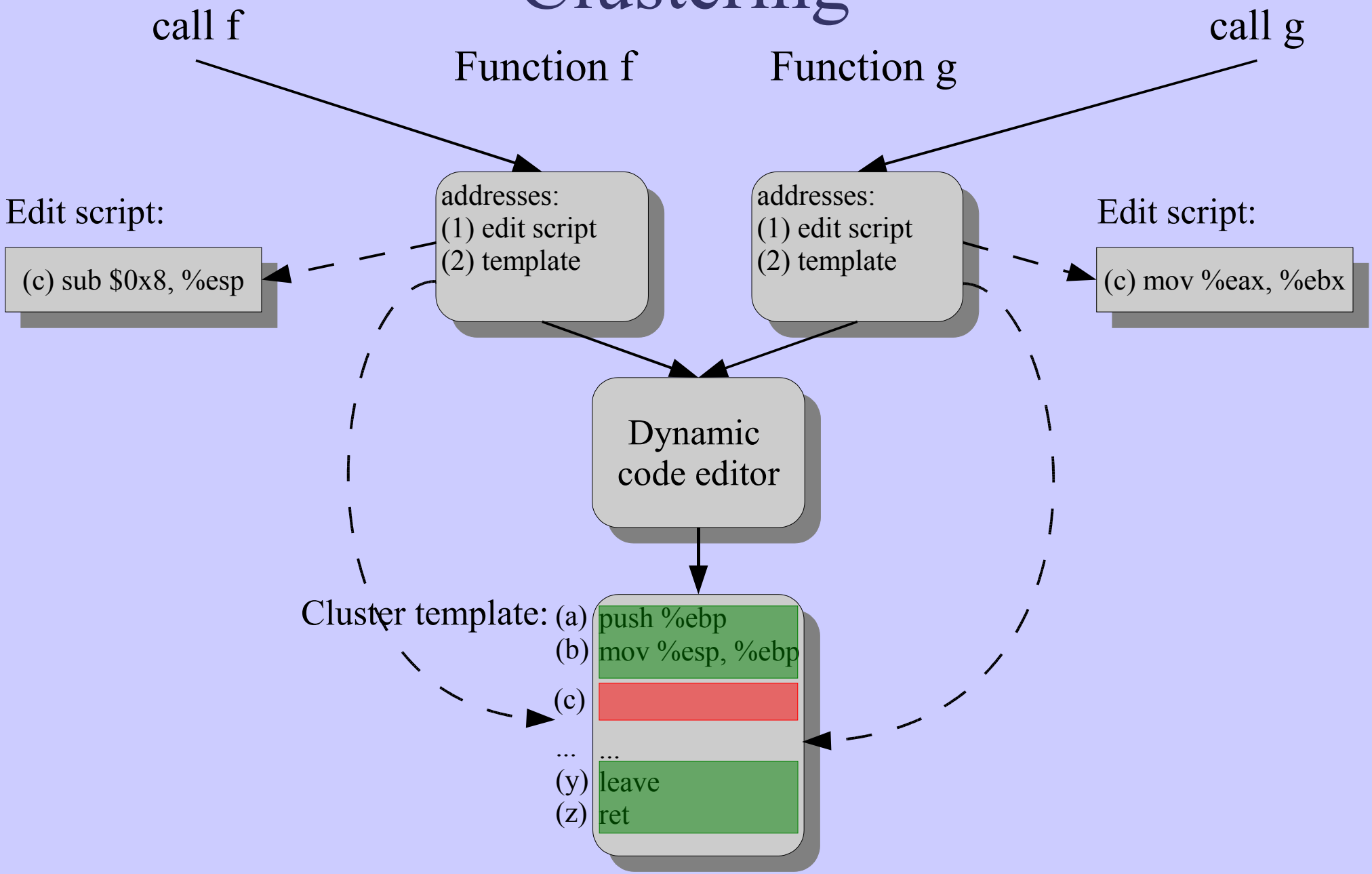

call f
Function f
Function g
call g
push %ebp
mov %esp, %ebp
sub $0x8, %esp
...
leave
ret
push %ebp
mov %esp, %ebp
mov %eax, %ebx
...
leave
ret
(c) sub $0x8, %esp
(c) mov %eax, %ebx


Cluster template:
(a)
(b)
(c)
...
(y)
(z)
push %ebp
mov %esp, %ebp
...
leave
ret

Clustering

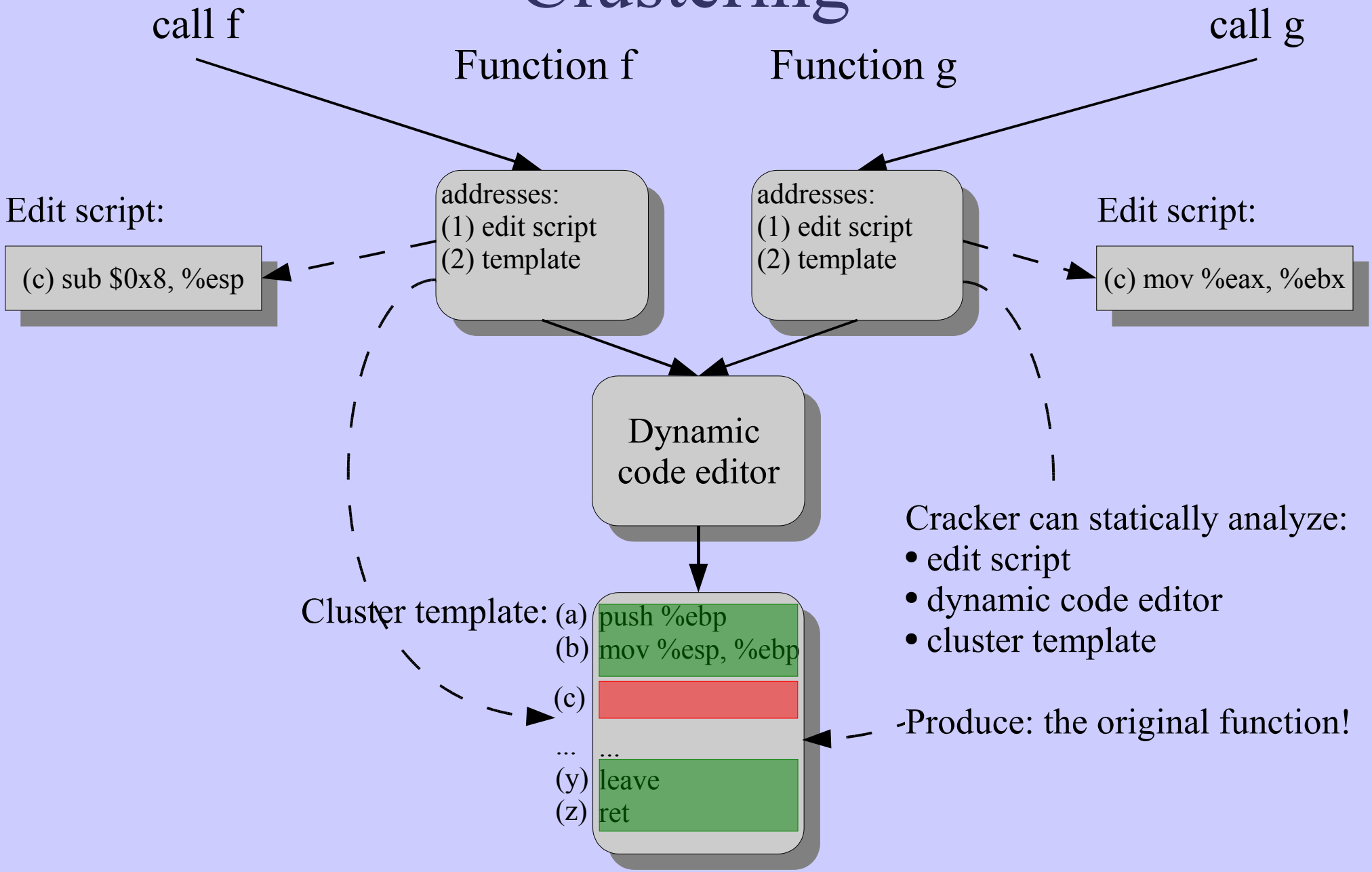

call f
Function f
Function g
call g
Edit script:
(c) sub $0x8, %esp
push %ebp
mov %esp, %ebp
sub $0x8, %esp
...
leave
ret
push %ebp
mov %esp, %ebp
mov %eax, %ebx
...
leave
ret
Edit script:
(c) mov %eax, %ebx
Cluster template:
(a) push %ebp
(b) mov %esp, %ebp
(c)
... ...
(y) leave
(z) ret

Clustering

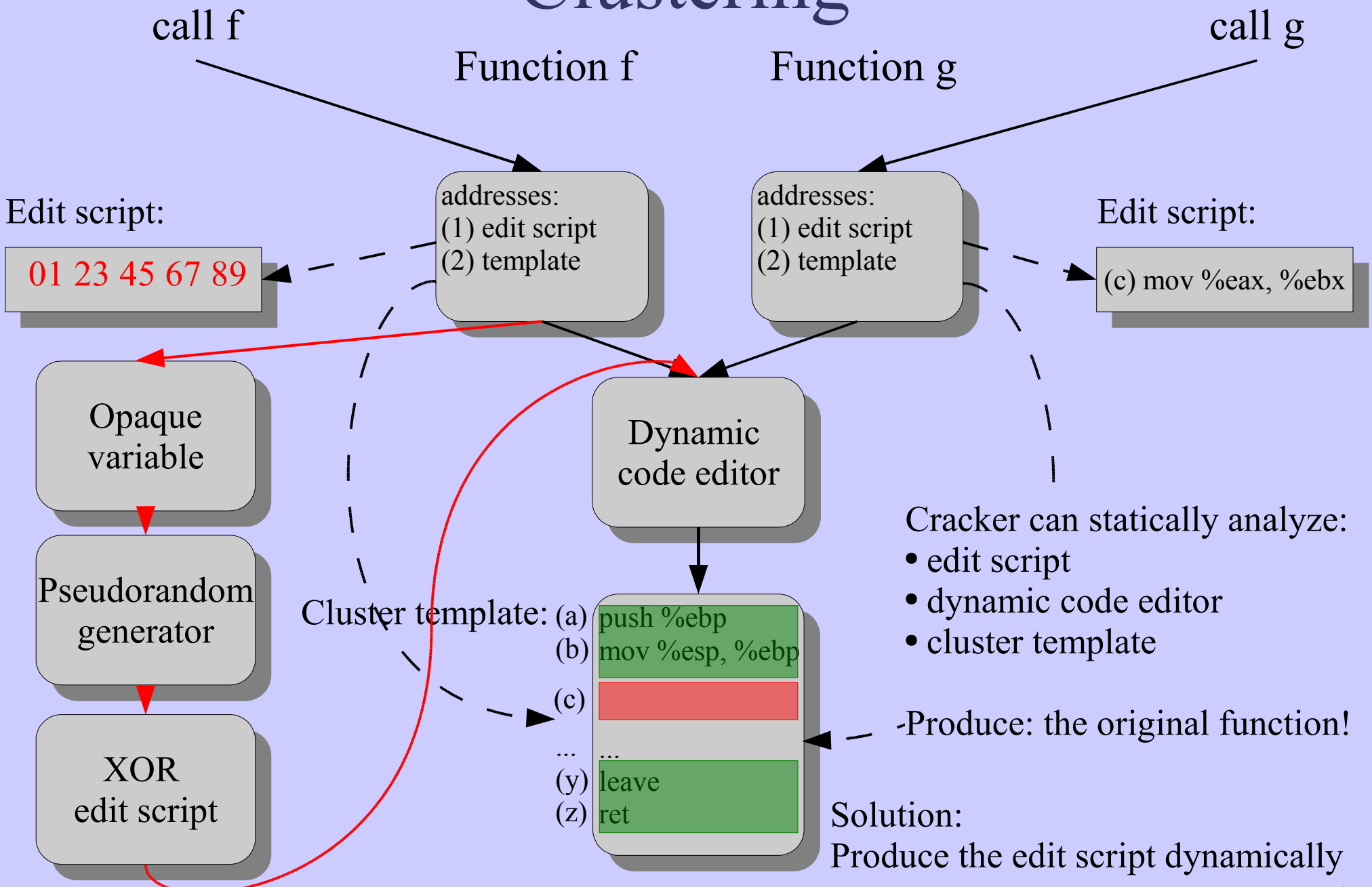


Dynamic production of function f and g

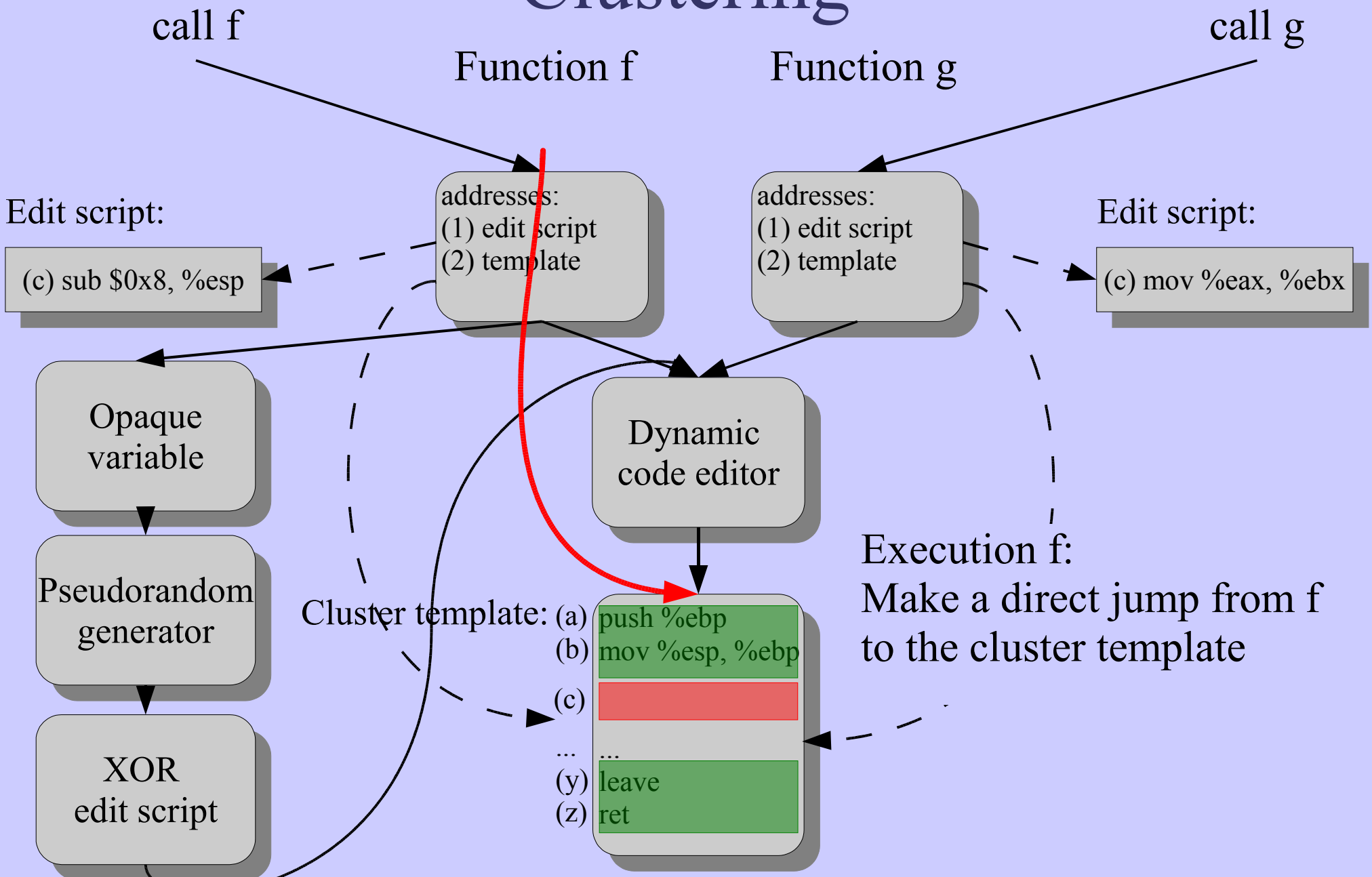
Clustering


call f
Function f
Function g
call g
Edit script:
(c) sub $0x8, %esp
addresses:
(1) edit script
(2) template
addresses:
(1) edit script
(2) template
Edit script:
(c) mov %eax, %ebx
Dynamic
code editor
Cluster template:
(a) push %ebp
(b) mov %esp, %ebp
(c)
...
(y) leave
(z) ret
Cracker can statically analyze:
• edit script
• dynamic code editor
• cluster template
Produce: the original function!

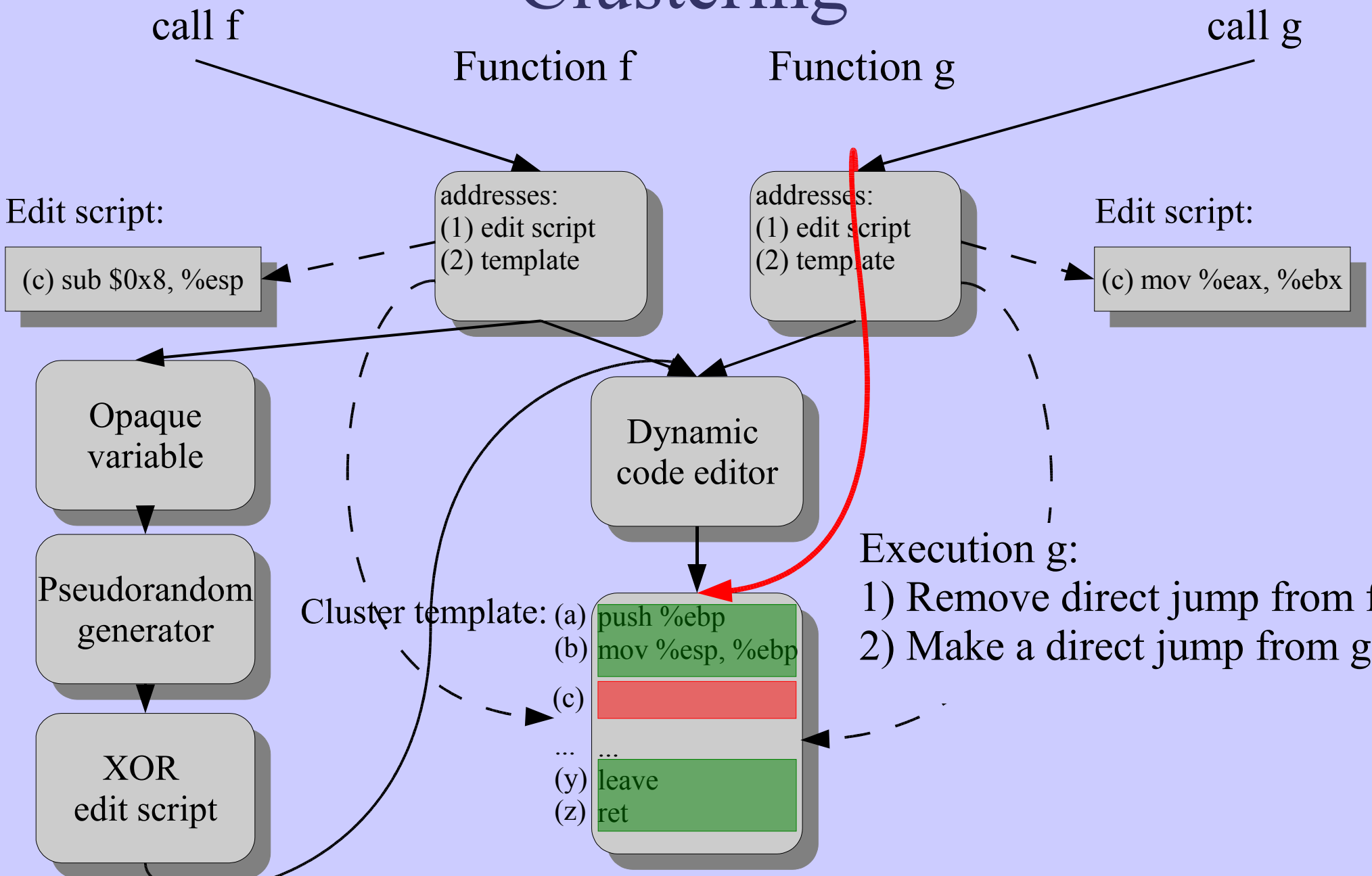
Clustering


call f
Function f
Function g
call g
Edit script:
01 23 45 67 89
addresses:
(1) edit script
(2) template
addresses:
(1) edit script
(2) template
Edit script:
(c) mov %eax, %ebx
Opaque variable
Pseudorandom generator
XOR edit script
Dynamic code editor
Cluster template:
(a) push %ebp
(b) mov %esp, %ebp
(c)
...
(y) leave
(z) ret
Cracker can statically analyze:
• edit script
• dynamic code editor
• cluster template
-Produce: the original function!
Solution:
Produce the edit script dynamically

Clustering


call f
Function f
Function g
call g
Edit script:
(c) sub $0x8, %esp
addresses:
(1) edit script
(2) template
addresses:
(1) edit script
(2) template
Edit script:
(c) mov %eax, %ebx
Opaque variable
Pseudorandom generator
XOR edit script
Dynamic code editor
Cluster template:
(a) push %ebp
(b) mov %esp, %ebp
(c)
...
(y) leave
(z) ret
Execution f:
Make a direct jump from f to the cluster template

Clustering

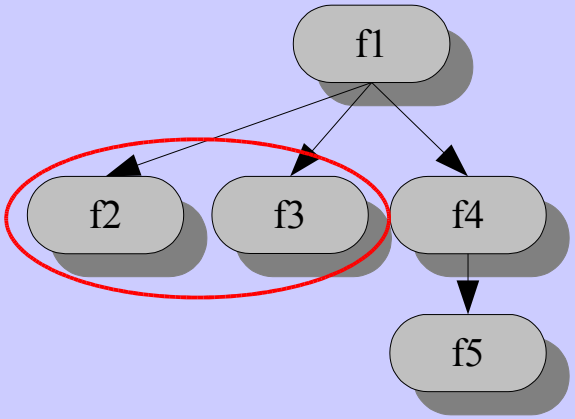

call f
Function f
Function g
call g
Edit script:
(c) sub $0x8, %esp
addresses:
(1) edit script
(2) template
addresses:
(1) edit script
(2) template
Edit script:
(c) mov %eax, %ebx
Opaque variable
Pseudorandom generator
XOR edit script
Dynamic code editor
Cluster template:
(a) push %ebp
(b) mov %esp, %ebp
(c)
...
(y) leave
(z) ret
Execution g:
1) Remove direct jump from f
2) Make a direct jump from g!

Clustering algorithm

Cluster: f2 and f3
Execution: f1 f2 f3 f2 f3 f2 f3 ...

Not good:
- Only two functions protected
- A lot of calls to dynamic code editor

callgraph of program P

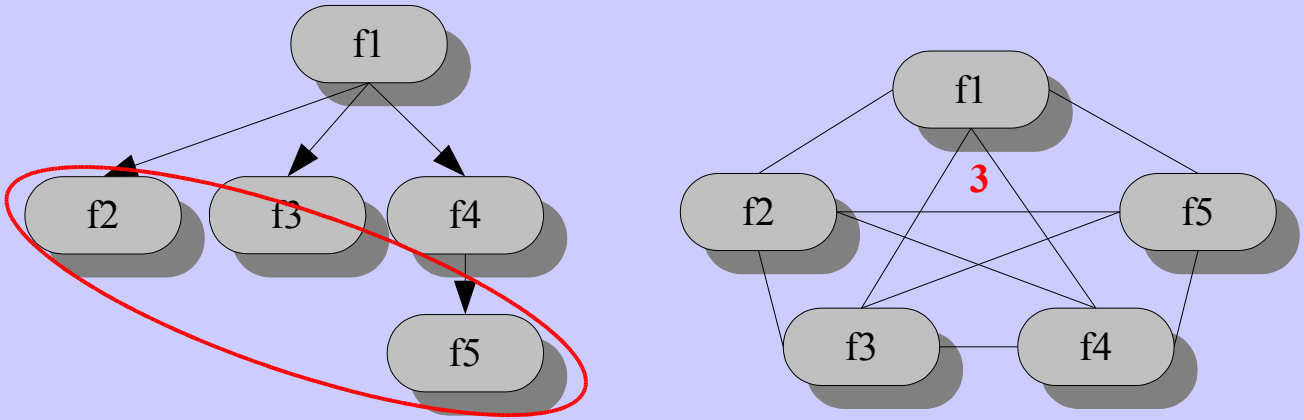


5 functions: f1, f2, f3, f4 and f5

Needed: A good clustering algorithm!

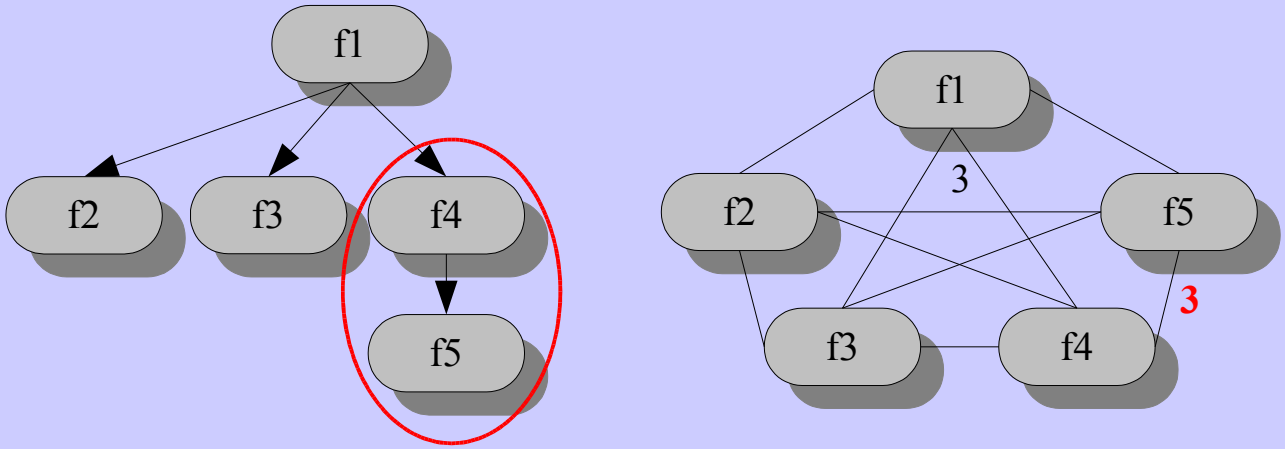
Clustering: Practical

Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1



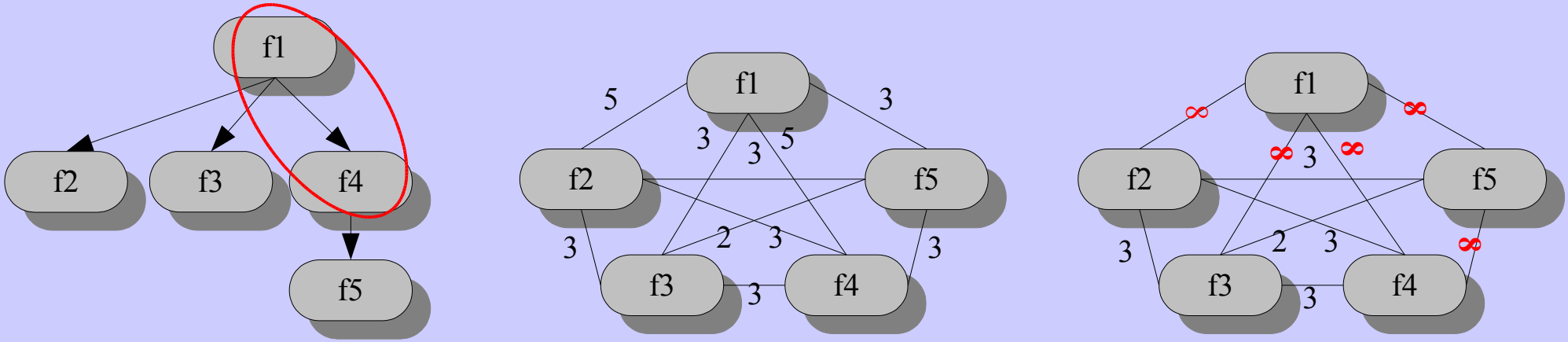
Clustering: Practical

Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1



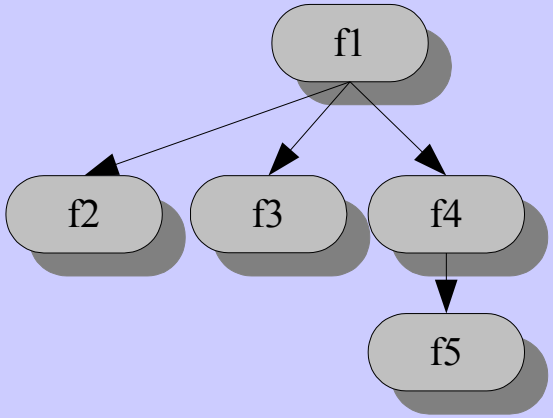
Clustering: Practical

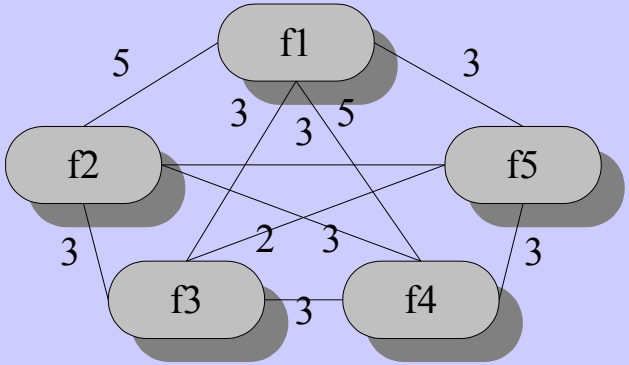
Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1

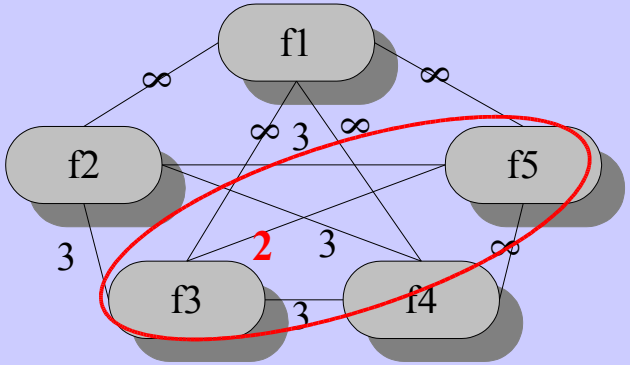


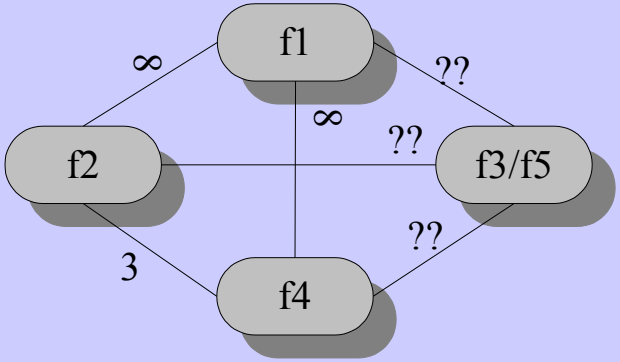
Clustering: Practical

Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1



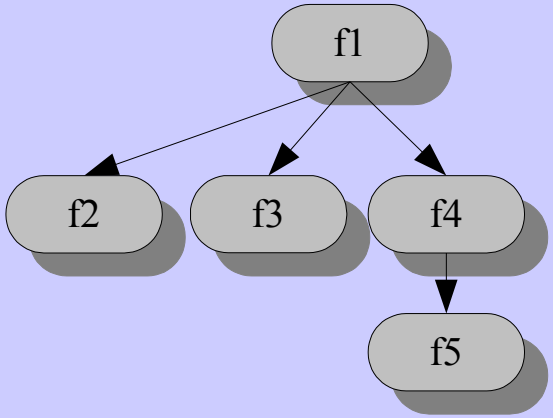


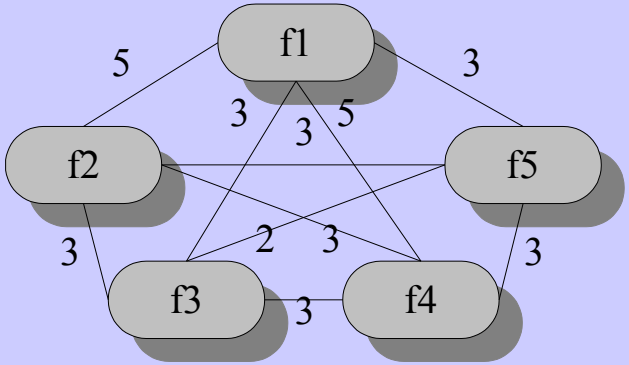


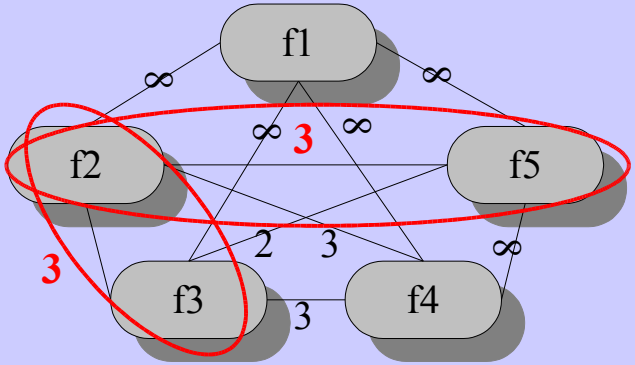


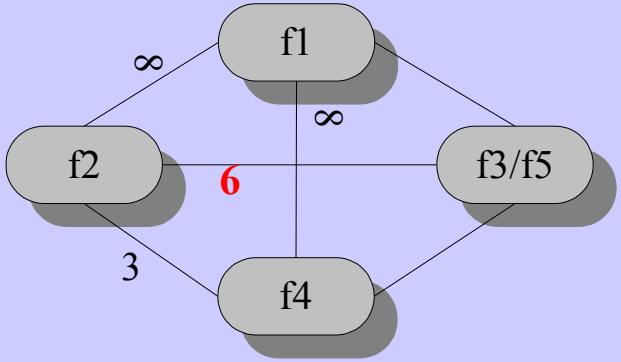
Clustering: Practical

Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1



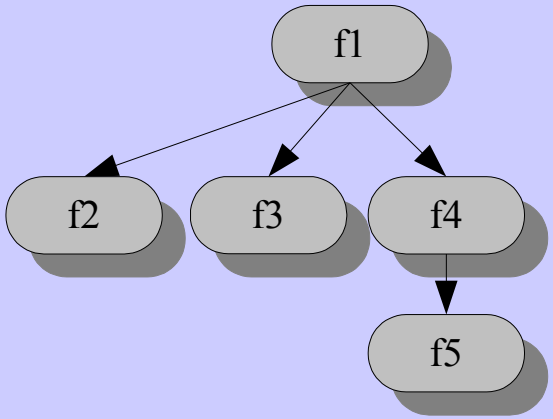


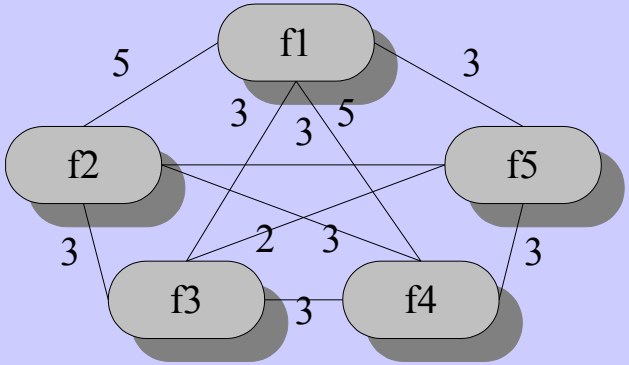


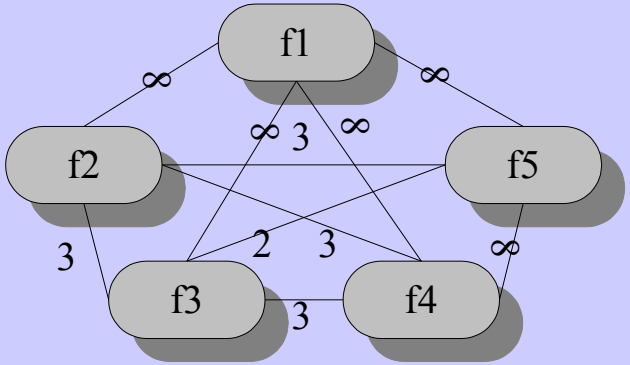


Clustering: Practical

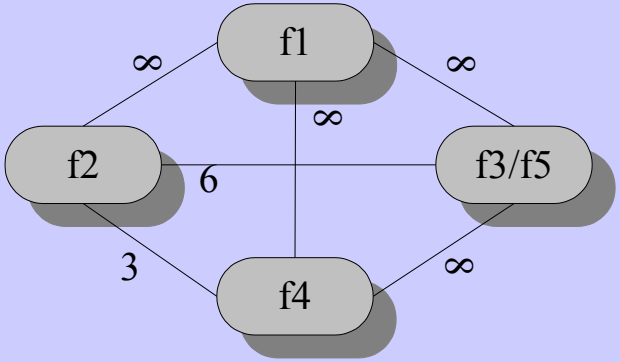
Training execution: f1 f2 f1 f4 f1 f3 f1 f4 f5 f4 f1 f2 f1

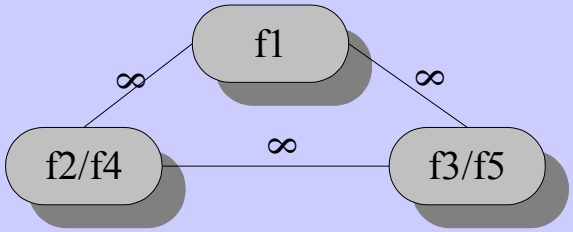






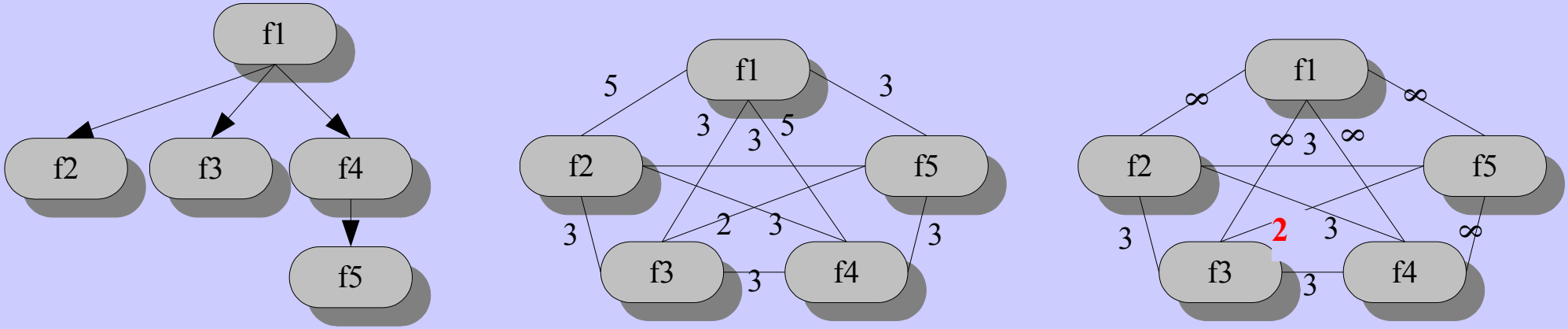
Tradeoff:
security ↔ slowdown





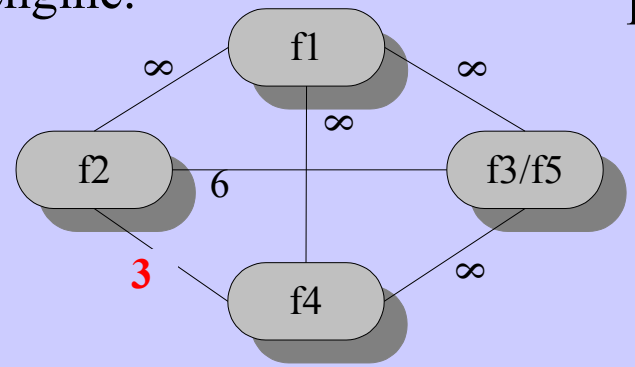
Clustering: Practical

Maximum dynamic code editor calls = 33

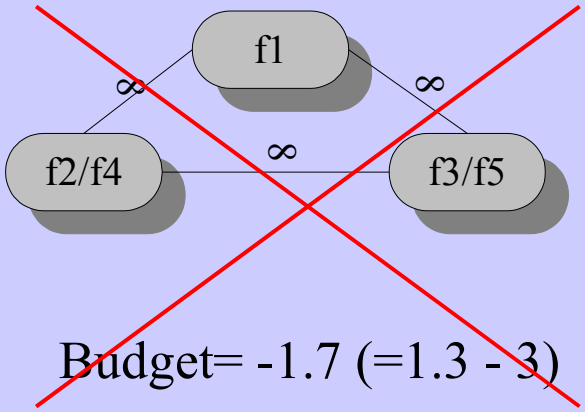


budget = fraction phi of the number of procedure calls n that can be preceded by a call to the editing engine.

Example: phi = 10% Budget = 3.3



Budget= 1.3 (=3.3 - 2)

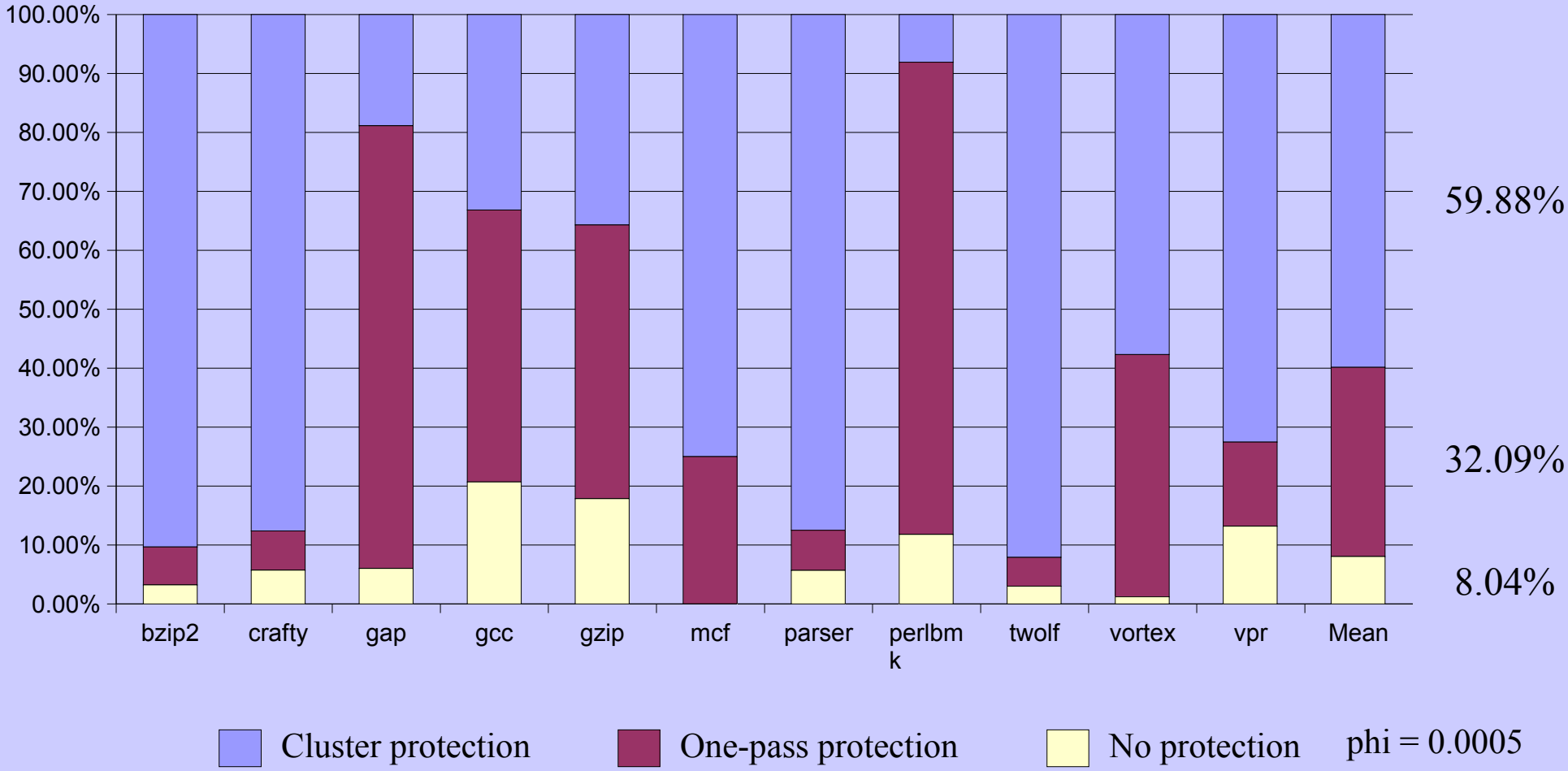


Budget= -1.7 (=1.3 - 3)

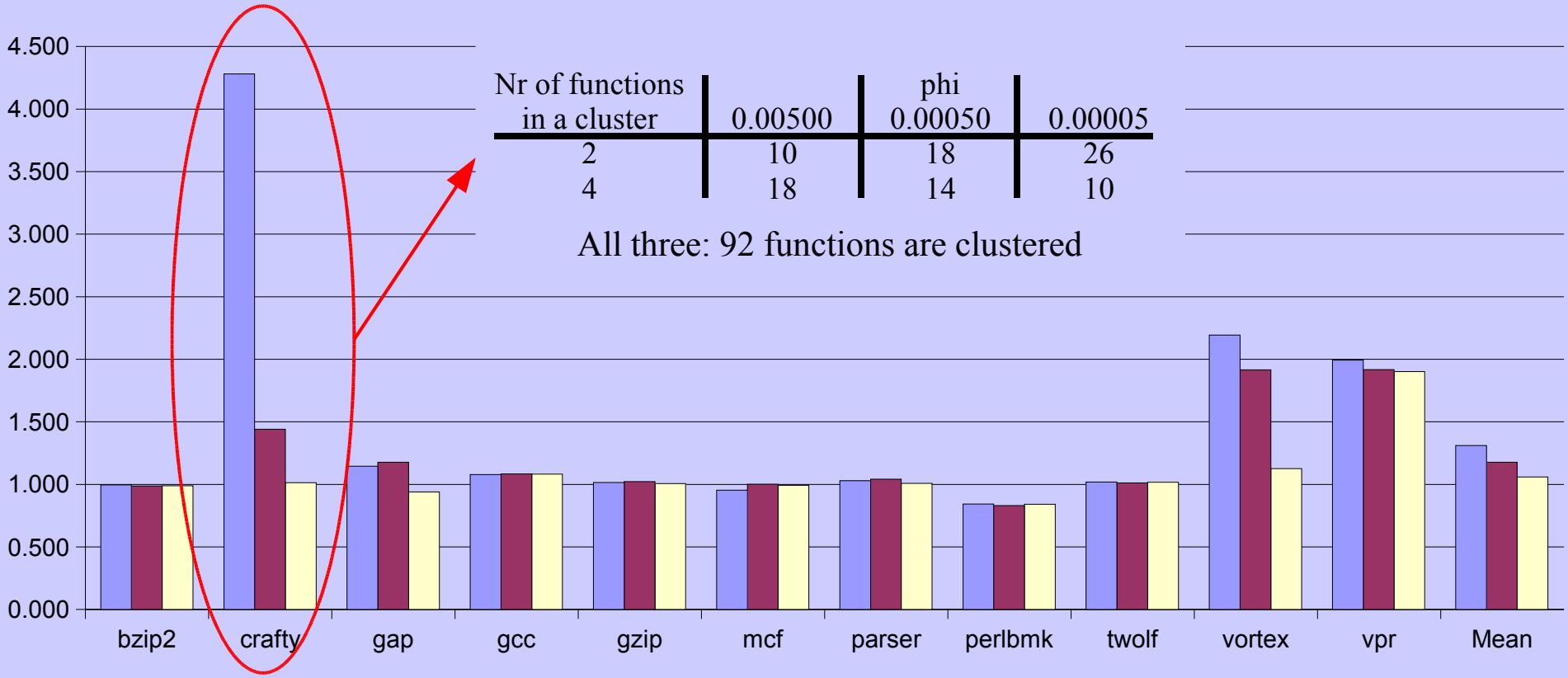
Security

- Disassembly:
  - Static: no longer possible
  - Dynamic: only code that is executed
- Tools for program analysis and reverse engineering cannot deal with dynamically mutating code
- Statically: as secure as the least secure component of opaque variables and pseudorandom generators

Number of procedures that can be protected



Execution time for different phi values



Nr of functions in a cluster	phi 0.00500	0.00050	0.00005
2	10	18	26
4	18	14	10

All three: 92 functions are clustered

Questions ?

Presentation: http://www.elis.ugent.be/~mmadou/home/papers/wisa05/wisa05paper.ps
Full Paper: http://www.elis.ugent.be/~mmadou/home/papers/wisa05/wisa05presentation.pdf

Bibtex:

```
@inproceedings{mmadouwisa05,
  title     =  "Software Protection through Dynamic Code Mutation",
  author    =  "Madou, Matias and Anckaert, Bertrand and Moseley, Patrick and Debray, Saumya and De Sutter, Bjorn
                and De Bosschere, Koen",
  booktitle =  "The 6th International Workshop on Information Security Applications (WISA 2005)",
  year      =  "2005",
   pages    =  "???",
   month    =  "August",
   publisher =  "Springer Verlag",
   journal  =  "Lecture Notes in Computer Science",
   volume   =  "LNCS"
}
```