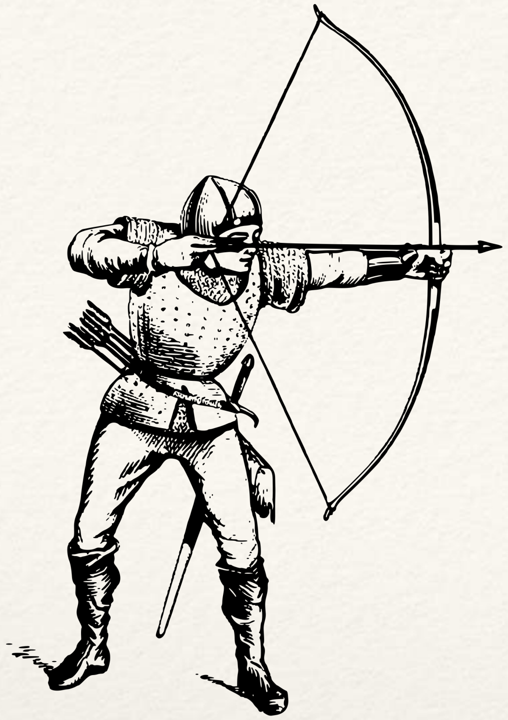
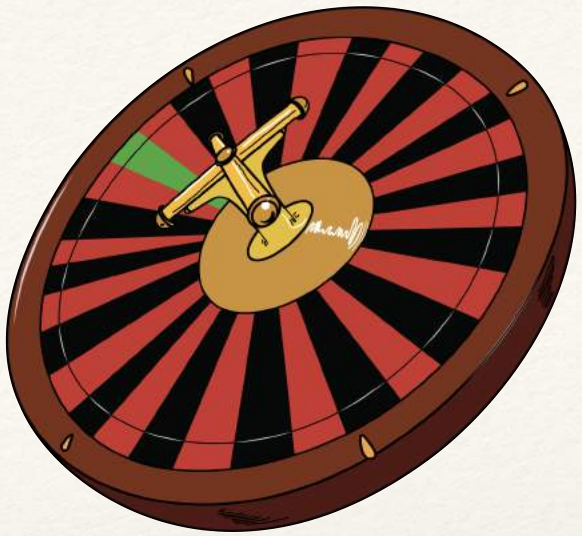




Reasoning about Probabilistic Defence Mechanisms against Remote Attacks

M. Ochoa, SUTD
S. Banescu, TUM
C. Disenfeld, U. Toronto
G. Barthe, IMDEA Software
V. Ganesh, U. Waterloo

IEEE Euro S&P - Paris, 2017

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Motivation

- Vulnerable code in online services is a threat despite several vanilla vulnerabilities are well known
  - C/C++ compiled without memory safety is fast.
  - Secure Coding is difficult.
  - Code verification is expensive.
  - Legacy code is difficult to maintain.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Moving target defences

- One popular idea is to raise the bar against exploitation:
  - By randomly altering memory layout (*ASLR*)
  - Forcing the guessing of some random key (*Canaries*)
  - Encrypting instructions (*ISR*)
  - etc.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Problem

- Despite convenience of such defences, guarantees provided by them are often unclear.
  - How can we spell out the assumptions made when designing such countermeasures?
  - Can we quantify their security guarantees in a formal sense?
  - Are guarantees better when countermeasures are composed?

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Challenges

Address	0	1	2	3	4	5	6	7	8	9
00000000	b3	2b	00	3a	35	ba	dd	66	57	7c
00000010	43	46	d1	31	a7	c5	4b	b8	2f	fe
00000020	24	79	23	dc	21	f6	2c	d4	18	2e
00000030	ab	ca	ed	af	02	61	51	0d	4e	ea
00000040	71	31	30	9c	c4	28	0e	b4	24	3d
00000050	62	96	d9	bf	f7	39	7e	43	90	98
00000060	e4	02	78	b3	a5	4f	5d	dc	69	75
00000070	1a	62	59	5a	9f	63	0b	07	95	91
00000080	38	8c	45	fb	9d	85	0c	fe	91	35

- What is the right abstraction level?
  - Too detailed: intractable
  - Too abstract: connection to real systems?
- Our goal: reach a compromise that can aid in the design of probabilistic countermeasures and their composition.
- Quantification is as meaningful as abstraction.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Crypto Proofs

- In modern cryptography:
  - Security arguments estimate the probability of a certain unwanted event, for instance:
    - Semantic security
    - Collisions in stream-ciphers
  - Unknown but computationally <u>bounded</u> adversary.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

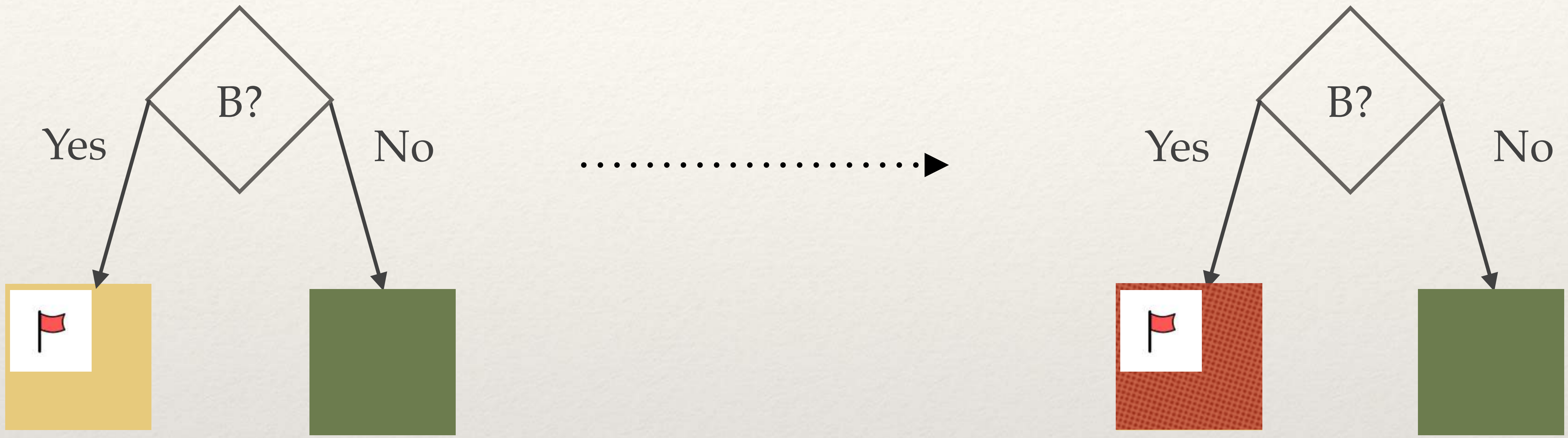
Crypto Proofs

- To achieve rigour in such arguments
  - Probabilistic programming languages are used:
    - Where $x \xleftarrow{\$} S$ denotes random assignment from set S.
  - Game hopping technique has been widely advocated (Shoup, Bellare, Rogaway).

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Proof strategy

Original game G1: Goal game G2:

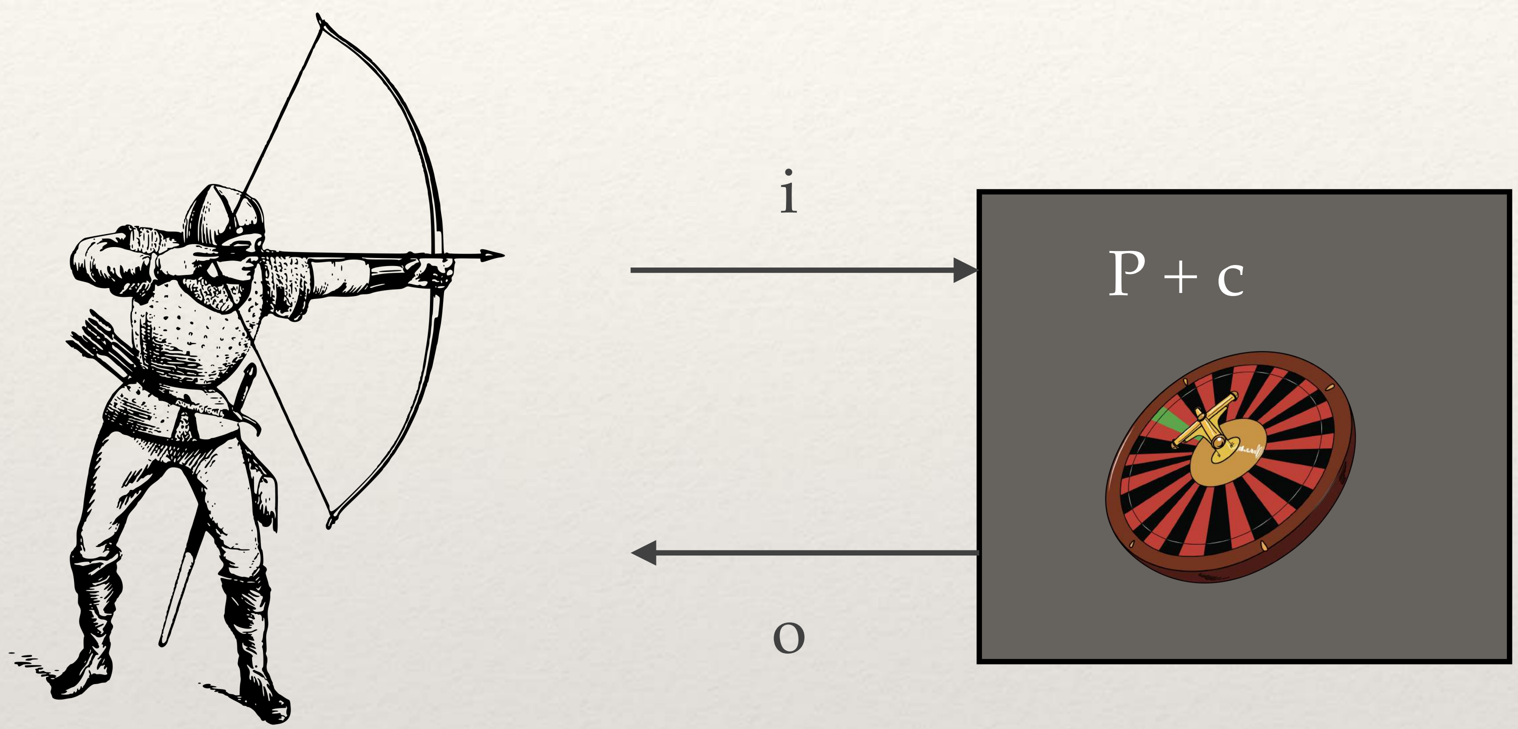


Pr[E] = ? Pr[E] = k

| Pr[E]_G1 - Pr[E]_G2 | <= Pr[B]

E = Event linked with security proof
B = Event that triggers “bad” behaviour
Fundamental lemma [Shoup ’04].

System model



Attacker's knowledge:

$$\forall\ \omega \in \Omega,\ \ [\![P]\!](\omega) = \text{crash}$$

Attacker's goal:

$$i \in \Omega(P) : [P + c](i) \neq \text{crash}$$

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Security Definition

We define an effective probabilistic countermeasure *c* if:

$$|\Pr[\mathcal{A}^{[P+c]} = i]| \leq \epsilon(n)$$

Where attacker performs *q* queries to program and *q* is polynomial on *n* and

$$i \in \Omega(P) : [P+c](i) \neq \mathsf{crash}$$

We abstract away from consequence of attack.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Ideal game

Ideal program execution:

$$\begin{array}{l} \textbf{Proc. } [\![P]\!](i) \\ \textsf{if } i \in \Omega(P) \textsf{ then} \\ \quad o \leftarrow \textsf{crash} \\ \textsf{else} \\ \quad o \leftarrow [P](i) \\ \textsf{return } o \end{array}$$

Such that:

$$\Pr[i \in \Omega(P) \wedge o \neq \textsf{crash}] = 0$$

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Unsafe execution

Real program execution:

$$
\begin{array}{l}
\textbf{Proc. } [P + \emptyset](i) \\
\textsf{If } i \in \Omega(P) \textsf{ then} \\
\quad ra \leftarrow i.\textsf{payload}[0] \\
\quad \boxed{\begin{array}{l} \textsf{If } ra \in \textsf{Valid then} \\ \quad o \leftarrow [\mathcal{M}(P, ra)] \end{array}} \\
\quad \textsf{else} \\
\quad\quad o \leftarrow \textsf{crash} \\
\textsf{else} \\
\quad o \leftarrow [P](i) \\
\textsf{return } o
\end{array}
$$

Such that:

$$\Pr[i \in \Omega(P) \wedge o \neq \textsf{crash}] = \Pr[i \in \Omega(P) \wedge i.\textsf{payload}[0] \in \textsf{Valid}]$$

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Example: Canaries

$$
\begin{array}{l}
\textbf{Proc. } [P + c](i) \\
\text{If } i \in \Omega(P) \text{ then} \\
\quad k \xleftarrow{\$} \{0,1\}^n \\
\quad ca \leftarrow i.\text{payload}[0] \\
\quad ra \leftarrow i.\text{payload}[1] \\
\quad \text{If } ca = k \text{ and } ra \in \text{Valid then} \\
\quad\quad o \leftarrow [\mathcal{M}(P, ra)] \\
\quad \text{else} \\
\quad\quad o \leftarrow \text{crash} \\
\text{else} \\
\quad o \leftarrow [P](i) \\
\text{return } o
\end{array}
$$

Where:

$$
\begin{aligned}
\Pr[E] &= \Pr[i \in \Omega(P) \wedge i.\text{payload}[0] = k \\
&\qquad \wedge\, i.\text{payload}[1] \in \text{Valid}] \\
&\leq \Pr[i \in \Omega(P) \wedge i.\text{payload}[0] = k] \\
&\leq \Pr[i.\text{payload}[0] = k] = \frac{1}{2^n}
\end{aligned}
$$

For q attempts: $\frac{q}{2^n}$ ← Polynomial in n

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Canaries: Game-based proof 1

Add bad · *After bad nothing matters* · *Remove bad*

Proc. $[P + c](i)$
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.\mathsf{payload}[0]$
  $ra \leftarrow i.\mathsf{payload}[1]$
  If $ca = k$ and $ra \in \mathsf{Valid}$ then
    $o \leftarrow [\mathcal{M}(P, ra)]$
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

$\mathbf{G_1}(i)$
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.\mathsf{payload}[0]$
  $ra \leftarrow i.\mathsf{payload}[1]$
  If $ca = k$ and $ra \in \mathsf{Valid}$ then
    $\boxed{bad \leftarrow true}$
    $o \leftarrow [\mathcal{M}(P, ra)]$
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

$\mathbf{G_2}(i)$
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.\mathsf{payload}[0]$
  $ra \leftarrow i.\mathsf{payload}[1]$
  If $ca = k$ and $ra \in \mathsf{Valid}$ then
    $\boxed{bad \leftarrow true}$
    $o \leftarrow$ crash
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

$\mathbf{G_3}(i)$
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.\mathsf{payload}[0]$
  $ra \leftarrow i.\mathsf{payload}[1]$
  If $ca = k$ and $ra \in$ V
    $o \leftarrow$ crash
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

Fundamental Lemma

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Canaries: Game-based proof 2

After bad nothing matters

Remove bad

Branch coalescing

Dead code elimination

0]
1]
$a \in$ Valid then
$a)]$

G₂(i)
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.$payload$[0]$
  $ra \leftarrow i.$payload$[1]$
  If $ca = k$ and $ra \in$ Valid then
    $\boxed{bad \leftarrow true}$
    $o \leftarrow$ crash
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

G₃(i)
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.$payload$[0]$
  $ra \leftarrow i.$payload$[1]$
  If $ca = k$ and $ra \in$ Valid then
    $o \leftarrow$ crash
  else
    $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

G₄(i)
If $i \in \Omega(P)$ then
  $k \xleftarrow{\$} \{0,1\}^n$
  $ca \leftarrow i.$payload$[0]$
  $ra \leftarrow i.$payload$[1]$
  $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

Proc. $[\![P]\!](i)$
if $i \in \Omega(P)$ then
  $o \leftarrow$ crash
else
  $o \leftarrow [P](i)$
return o

lamental Lemma

SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Composition: ASLR and Canaries

$$
\begin{array}{l}
\textbf{Proc. } [P + c](i) \\
\text{If } i \in \Omega(P) \text{ then} \\
\quad \boxed{k_1 \xleftarrow{\$} \{0,1\}^n} \\
\quad \boxed{k_2 \xleftarrow{\$} \{0,1\}^m} \\
\quad ca \leftarrow i.\text{payload}[0] \\
\quad ra \leftarrow i.\text{payload}[1] \\
\quad \text{If } ca = k_1 \text{ then} \\
\quad\quad \text{If } ra \in \Pi_{k_2}(\text{Valid}) \text{ then} \\
\quad\quad\quad o \leftarrow [\mathcal{M}(P, ra)] \\
\quad\quad \text{else} \\
\quad\quad\quad o \leftarrow \text{crash} \\
\quad \text{else} \\
\quad\quad o \leftarrow \text{crash} \\
\text{else} \\
\quad o \leftarrow [P](i) \\
\text{return } o
\end{array}
$$

Where:

$$
\begin{aligned}
\Pr[i \in \Omega(P) \wedge o \neq \text{crash}] &\leq \Pr[i.\text{payload}[0] = k_1 \\
&\qquad \wedge \ i.\text{payload}[1] \in \Pi_{k_2}(\text{Valid})] \\
&\leq \frac{1}{2^n} \cdot \frac{|\text{Valid}|}{2^m}
\end{aligned}
$$

Bounds

Composition	**n-bit Architecture**	**32-bit Architecture**	**64-bit Architecture**
ASLR⊗PointGuard	$\frac{q \cdot \lvert\mathsf{Valid}\rvert}{2^n - q}$	1	2^{-23}
ASLR⊗ISR	$\frac{q \cdot \lvert\mathsf{Valid}\rvert}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{ISA}\rvert}{2^n - r}$	2^{-10}	2^{-75}
PointGuard⊗ISR	$\frac{q \cdot \lvert\mathsf{Valid}\rvert}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{ISA}\rvert}{2^n - r}$	2^{-10}	2^{-75}
Canary⊗ASLR	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r}$	2^{-22}	2^{-87}
Canary⊗PointGuard	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r}$	2^{-22}	2^{-87}
Canary⊗ISR	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{ISA}\rvert}{2^n - r}$	2^{-26}	2^{-91}
ASLR⊗PointGuard⊗ISR	$\frac{q \cdot \lvert\mathsf{Valid}\rvert}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{ISA}\rvert}{2^n - r}$	2^{-10}	2^{-75}
Canary⊗ASLR⊗PointGuard	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r}$	2^{-22}	2^{-87}
Canary⊗ASLR⊗ISR	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r} \cdot \frac{t \cdot \lvert\mathsf{ISA}\rvert}{2^n - t}$	2^{-42}	2^{-139}
Canary⊗PointGuard⊗ISR	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r} \cdot \frac{t \cdot \lvert\mathsf{ISA}\rvert}{2^n - t}$	2^{-42}	2^{-139}
Canary⊗ASLR⊗PointGuard⊗ISR	$\frac{q}{2^n - q} \cdot \frac{r \cdot \lvert\mathsf{Valid}\rvert}{2^n - r} \cdot \frac{t \cdot \lvert\mathsf{ISA}\rvert}{2^n - t}$	2^{-42}	2^{-139}

$q = 2^{25}$ ~ 9 days of queries

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Bounds

Composition	n-bit Architecture	32-bit Architecture	64-bit Architecture	128-bit Architecture
ASLR⊗PointGuard	$\frac{q\cdot\lvert\mathsf{Valid}\rvert}{2^n-q}$	1	2^{-23}	2^{-87}
ASLR⊗ISR	$\frac{q\cdot\lvert\mathsf{Valid}\rvert}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{ISA}\rvert}{2^n-r}$	2^{-10}	2^{-75}	2^{-203}
PointGuard⊗ISR	$\frac{q\cdot\lvert\mathsf{Valid}\rvert}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{ISA}\rvert}{2^n-r}$	2^{-10}	2^{-75}	2^{-203}
Canary⊗ASLR	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}$	2^{-22}	2^{-87}	2^{-215}
Canary⊗PointGuard	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}$	2^{-22}	2^{-87}	2^{-215}
Canary⊗ISR	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{ISA}\rvert}{2^n-r}$	2^{-26}	2^{-91}	2^{-219}
ASLR⊗PointGuard⊗ISR	$\frac{q\cdot\lvert\mathsf{Valid}\rvert}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{ISA}\rvert}{2^n-r}$	2^{-10}	2^{-75}	2^{-203}
Canary⊗ASLR⊗PointGuard	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}$	2^{-22}	2^{-87}	2^{-215}
Canary⊗ASLR⊗ISR	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}\cdot\frac{t\cdot\lvert\mathsf{ISA}\rvert}{2^n-t}$	2^{-42}	2^{-139}	2^{-331}
Canary⊗PointGuard⊗ISR	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}\cdot\frac{t\cdot\lvert\mathsf{ISA}\rvert}{2^n-t}$	2^{-42}	2^{-139}	2^{-331}
Canary⊗ASLR⊗PointGuard⊗ISR	$\frac{q}{2^n-q}\cdot\frac{r\cdot\lvert\mathsf{Valid}\rvert}{2^n-r}\cdot\frac{t\cdot\lvert\mathsf{ISA}\rvert}{2^n-t}$	2^{-42}	2^{-139}	2^{-331}

$q = 2^{25}$ ~ 9 days of queries

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Side-channels

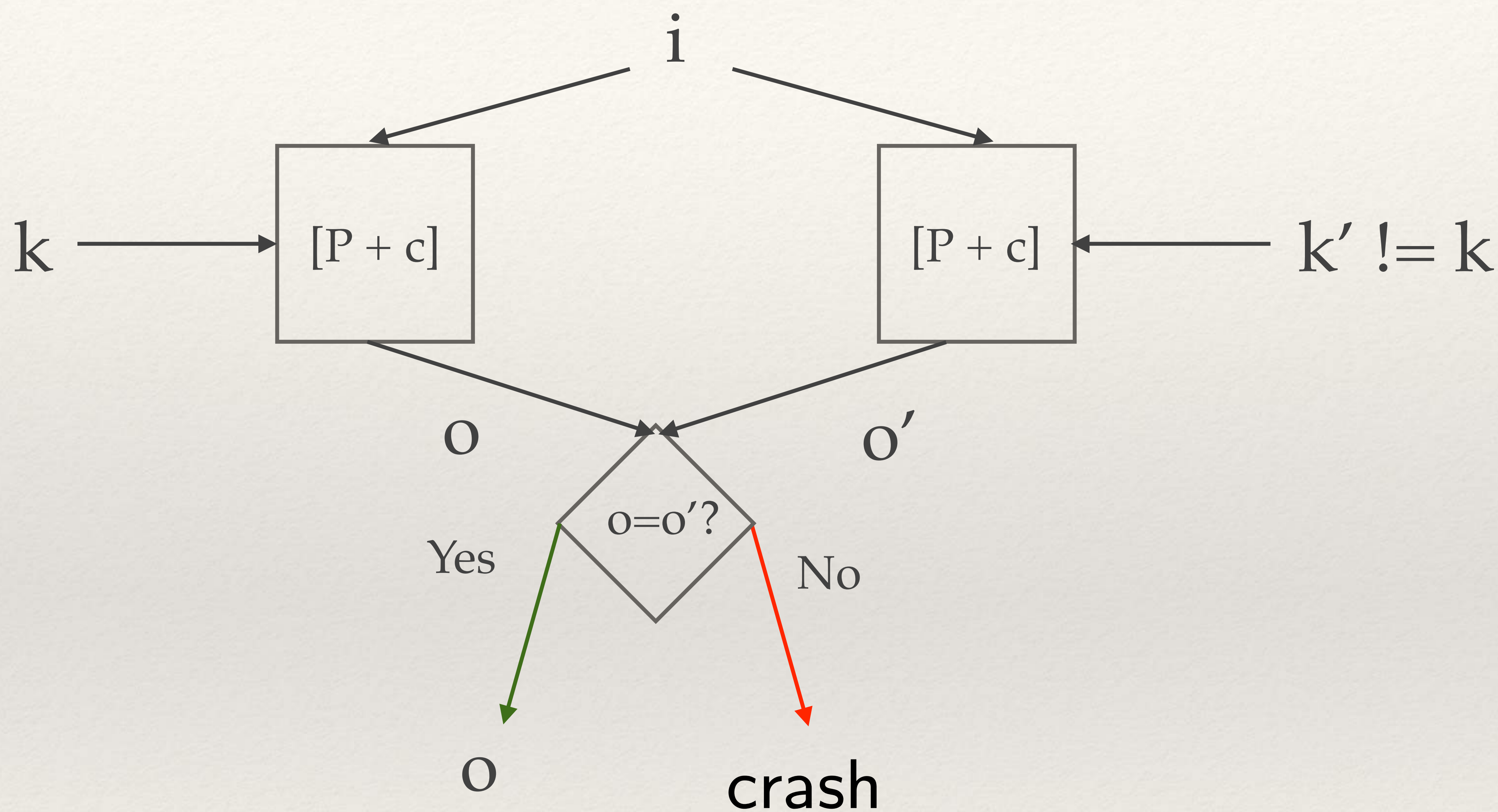
If leakage is known, we can plug it in into our bounds, for instance for ASLR:

$$\Pr[i \in \Omega(P) \wedge o \neq \mathsf{crash}] \leq \frac{q_2 \cdot |\mathsf{Valid}|}{2^{n-\lambda} - q_2}$$

However in general it is difficult to foresee all side-channels.
Can we <u>close</u> them?

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Replicas



Surprisingly: $$\Pr[\mathcal{A}^{\mathsf{SME}([P+c])} = i] = 0$$

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Conclusions

- Presented a framework to reason about probabilistic defences against remote memory safety exploitation using the game-hopping technique.
- Showed how replicas can be used to close leakage dynamically.
- In the future we will apply our framework to other classes of probabilistic defences.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Questions?

Acknowledgements: Clipart take from opencliparts.org