FORMAL MODELLING AND VERIFICATION OF NAND FLASH MEMORY

P.Abhinav Kumar

FORMAL MODELLING AND VERIFICATION OF NAND FLASH MEMORY

by

P.Abhinav Kumar

Under the supervision

of

Dr. Srinivas Pinisetty

Thesis is submitted to the

Indian Institute of Technology Bhubaneswar

for award of the degree

of

Bachelor of Technology

SCHOOL OF ELECTRICAL SCIENCES

INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

MAY 2020

APPROVAL OF THE VIVA-VOCE BOARD

Date: 27^{th} May 2020

Certified that the thesis entitled **"Formal Modelling and Verification of NAND Flash Memory "**, submitted by **Mr. P.Abhinav Kumar** to the Indian Institute of Technology Bhubaneswar, for the award of the degree Bachelor of Technology has been accepted by the examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Supervisor) (External Examiner)

(Internal Examiner I) (Internal Examiner II)

CERTIFICATE

Certified that the thesis entitled **"Formal Modelling and Verification of NAND Flash Memory"**, submitted by **Mr. P.Abhinav Kumar** to the Indian Institute of Technology Bhubaneswar, for the award of the degree Bachelor of Technology has been accepted by the examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

Date:

Dr. Srinivas Pinisetty
(Supervisor)

DECLARATION

I certify that

1. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have followed the guidelines provided by the Institute in writing the thesis.
4. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
6. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Signature of the Student

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my thesis Supervisor, Dr.Srinivas Pinisetty for his vital support and assistance.It has been a wonderful learning experience with his constant support and guidance towards the right direction inspite of him being busy with his own research and teaching. His knowledge and constant support helped me towards the goal.

I would also like to show my gratitude towards Shivani Tripathy for her valuable suggestions and also guiding me throughout my work. I would finally like to thank my friends and family for extending their moral support while I was working on the project.

I will use all the knowledge and skills gained while working on this project towards the best possible way.

P.Abhinav Kumar

ABSTRACT

RBlocker provides full protections against all possible ransomware attacks by delaying every data deletion until no attack is confirmed. To reduce storage overheads of the delayed deletion,RBlocker employs a time-out based backup policy. Based on the fact that ransomware must store encrypted version of target files, early deletions of obsolete data are allowed if no encrypted write was detected for a short interval.As part of the project we develop formal models for Time out based Invalidator and additional state machine for page status.

The existing SSD-level solutions are insufficient either to be widely used in practice. FlashGuard , which provides full protections from ransomware attacks, suffers from a high space overhead of maintaining backup data owing to its conservative backup policy.The high space overhead makes FlashGuard almost impossible to be used. SSD-Insider addresses the limitations of FlashGuard by leveraging a ransomware detection algorithm on the storage side. By monitoring , it is able to detect whether a host is under a ransomware attack or not within 10 seconds. This early detection enables SSD-Insider not to maintain backup data for a long time, which mitigates the high space overhead problem.This gives need for development of ransom blocker which does not backup overly neither negligibly.

Contents

Chapter 1

Introduction

Ransomware encrypts user files and demands a ransom from a user for an encryption key to access the files.To address the potential vulnerability and backup overheads of host-level defense schemes, storage-level solutions have been proposed recently.Various anti-ransomware solutions try to protect user files by detecting ransomware before the ransomware runs and/or by backing up files on remote or local storage. However, since most antivirus programs run in a host system as a user application, they are vulnerable to evasion attacks with root privileges. Backing up original files cannot be a reliable solution either because a backup system itself can be infected . Moreover, creating a backup for recovery can cause significant overheads, which may degrade user experience.

1.1 Overview of the Area

Unlike the existing SSD-level solutions, RBlocker examines the content of incoming data for detecting ransomware attacks. Specifically, at run time, RBlocker determines whether incoming data is encrypted. Since encrypting victim files is an invariant step of all ransomware attacks, if an incoming write is not encrypted, we can guarantee that the data is not infected by ransomware. If encrypted data is identified, RBlocker delays the deletion of its original data by excluding them from the GC process.

RBlocker cannot tell which deleted data are the original data of the encrypted data. RBlocker overcomes this problem by employing a time-out based backup policy. When a file is deleted, RBlocker delays a deletion request by a given time-out threshold t

. After t time units, RBlocker checks if there were any encrypted writes in the past t units. If no encrypted writes were present, RBlocker concludes that it is safe to process the deletion request. If there were encrypted writes, RBlocker assumes that there was a ransomware attack and does not process the deletion request until no attack is guaranteed.

1.2 Problem Statement

NAND flash memory is a type of non- volatile storage technology that does not require power in order to retain data.The technology is used in common storage devices such as solid state drives and memory cards NAND flash memory is written and read in blocks that are smaller than the device.Globally, NAND Flash consumption has exploded over the last 5 years, and new products, such as NAND solid-state drives, are now making significant inroads into enterprise computing devices, from notebooks, desktops, workstations and servers.

Develop a new model for the controller of NAND flash-based SSDs and verify the correctness of the model.Within the SSD controller, there are important firmware modules like garbage collection agent, wear leveling agent and address translation agents. We want to model these agents.UPPAAL Tool is used for Formal Modelling and verification of NAND Flash Memory.

Chapter 2

Literature Review

2.1 Timed Automata

A real-time system is any information processing system which has to respond to externally generated input stimuli within a finite and specified period.The correctness depends not only on the logical result but also the time it was delivered.There are four types of Real Time Systems namely Hard real-time systems,Soft real-time systems,Firm real-time systems,Weakly hard real-time systems.Functional requirements (Operation of the system and their effects) and nonfunctional requirements(timing constraints) must be precisely defined and together used to construct the specification of the systemA specification is a mathematical statement of the properties to be exhibited by a system.

Timed automata is a theory for modeling and verification of real time systems. A timed automaton is a finite state Buchi automaton extended with a set of real-valued variables modeling clocks. Constraints on the clock variables are used to restrict the behavior of an automaton.Timed automaton may be considered as an abstract model of a timed system.

The main purpose of a model-checker is to verify the model w.r.t. a requirement specification. Like the model, the requirement specification must be expressed in a formally well-defined and machine readable language.

A state formula is an expression that can be evaluated for a state without looking at the behaviour of the model.Reachability properties are the simplest form of properties. They ask whether a given state formula, phi, possibly can be satisfied by

any reachable state.Reachability properties are often used while designing a model to perform sanity checks.Safety properties are on the form that something bad will never happen.Liveness properties are of the form that something will eventually happen.

2.2 Garbage Collection Policy in NAND Flash

Flash memory has the write-once property and cannot be directly used as an ideal block device, since a written page cannot be updated unless its residing block is erased.The selection of the victim blocks for garbage collection may considerably affect the lifetime, because each block endures only a limited number of erases before it is worn-out.It thus becomes a crucial design issue to ensure that all the blocks are evenly utilized and erased referred to as wear levelling .The garbage collection and wear leveling designs have been recognized as the key management facilities of flash memory.

2.3 Overview of UPPAAL Tool Kit

UPPAAL is a toolbox for modelling, simulating and verifying real-time systems. Appropriate for systems that can be modelled as a network of timed automata.

2.3.1 Locations in UPPAAL

In Normal Location , Time can pass as long as the invariant is satisfied , When the invariant becomes false the location must be exited. In Urgent Location ,there is no delay in transition. In Committed Location ,there is no delay in the transition. In a composition the transition out of the committed location must be exited first if more than one transition is enabled.

Uppaal model can check for invariant and reachability properties ; Whether certain combinations of locations and constraints on variables (clock and integer) are reachable.It also monitor automata,adds debugging information and checks reachability.It generates diagnostic trace

2.4 Understanding of UPPAAL Simulation with example

In the first activity, we assume that there are two cars that simultaneously search for parking space. The cars can move forward and backwards and their sensors will query a parallel automaton that models the street layout. Upon finding an available parking space they will move forward to reach the end of the available parking space and perform parallel reverse parking. Afterwards (after a fixed or arbitrary delay) they will move out of the parking space and move forward to the end of the street. For simplicity, assume that the street is 5 meters long, it has 2 suitable parking spaces (they may be fixed or randomly assigned in your model) and each car needs 1 meter space for parking. The only way for parallel automata to communicate is through shared channels; no other globally shared variables are allowed.

2.4.1 Correctness Criteria for Beginners

1. There is no state of deadlock when both cars end up at the end of the street.
2. Only one car can park in the first parking space at a time.
3. The cars should not be able to drive longer than 5 meters street.

2.4.2 Solution

1. Project Declarations

   chan park,unpark,move,continue move,end of street; bool second car parked = false ; bool first car parked = false ;

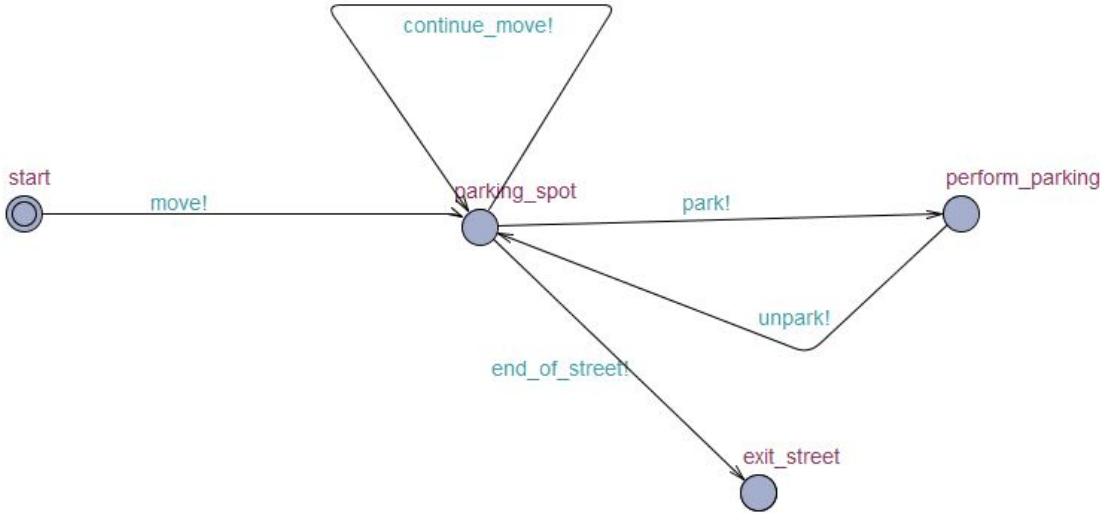
2. System Declarations

   carOne = Car(); carTwo = Car(); leftLane = Lane(); rightLane = Lane();

   system carOne,carTwo,leftLane,rightLane;

3. Car Template

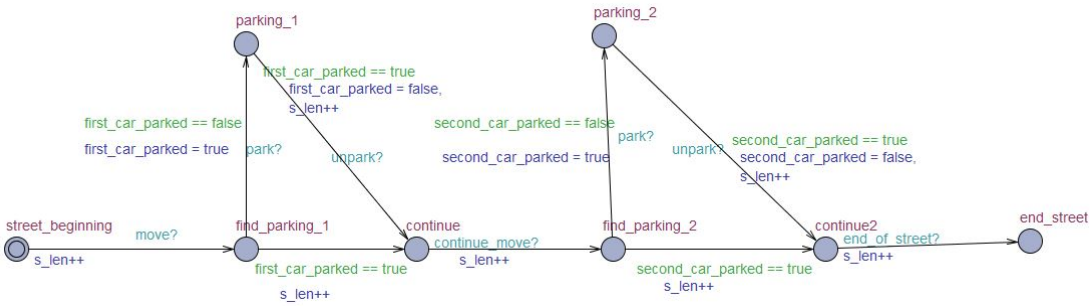
   Declarations :clock wait , int position = 0 ,bool parked = false



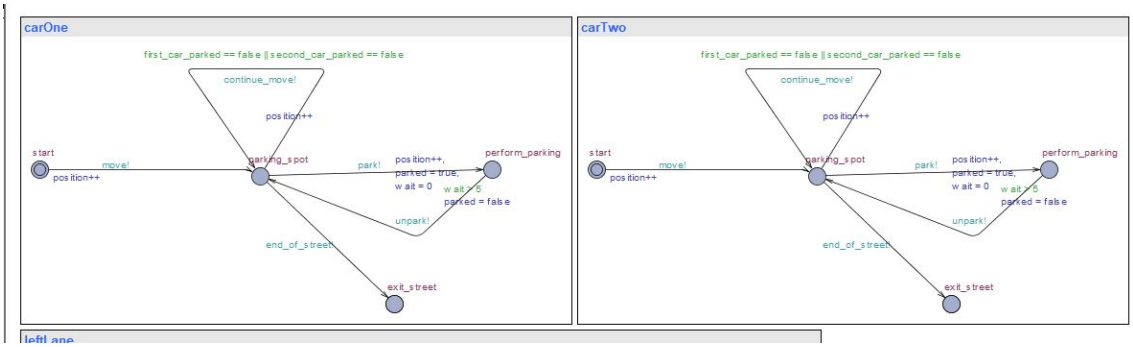
4. Lane Template :

   Declarations :

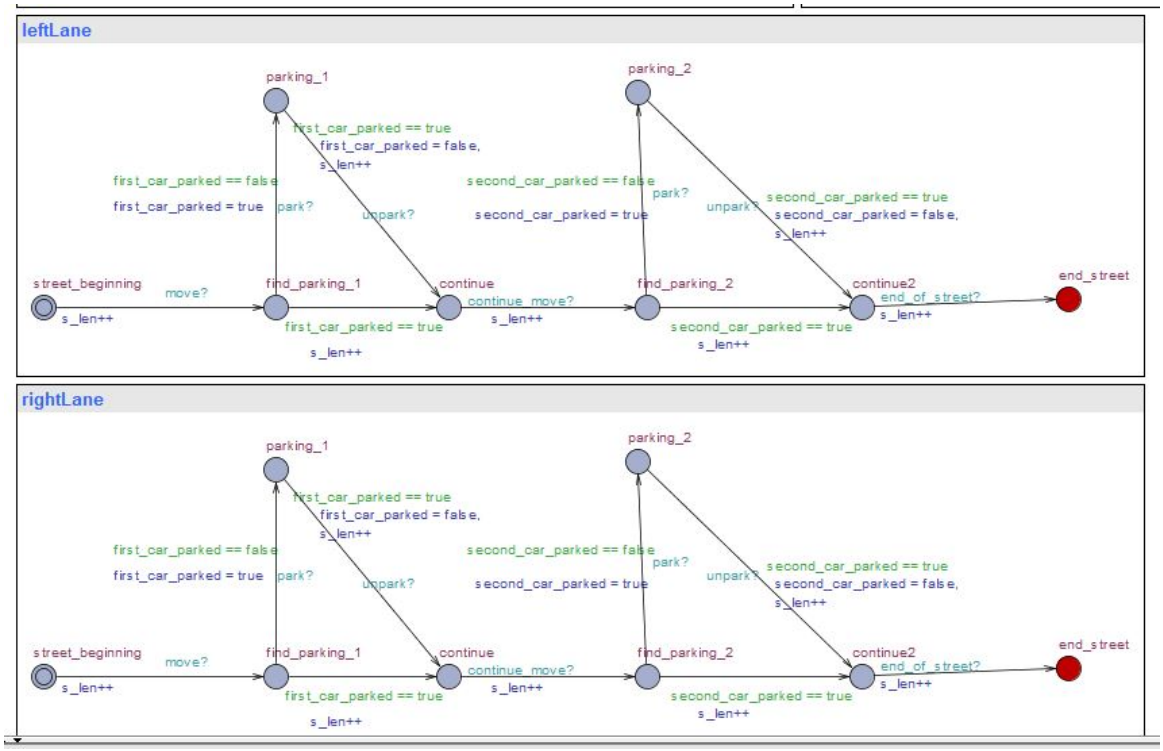
   int s len = 5 street = 5.

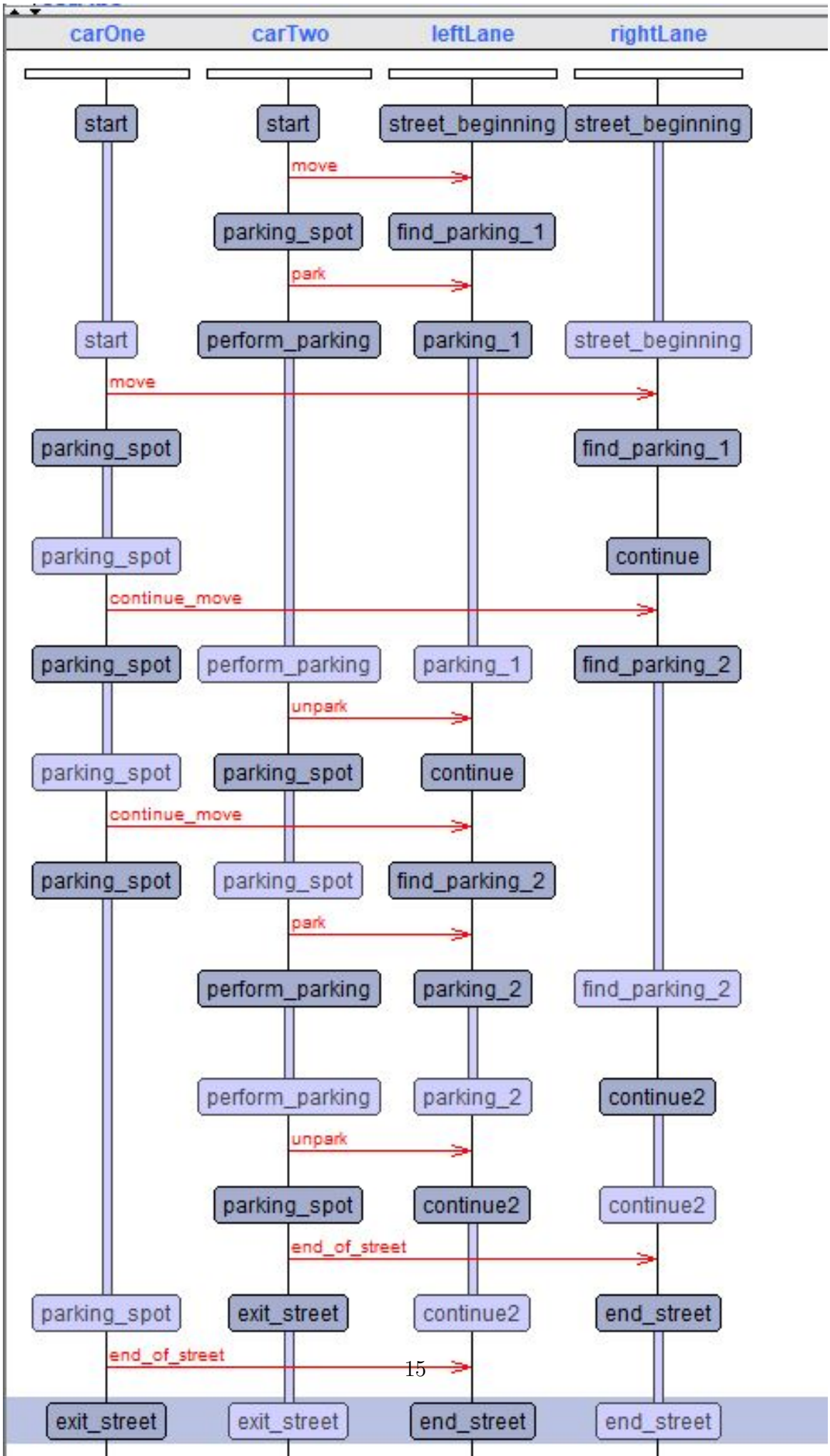


5. After Final Simulation:



6. Stages of simulation


leftLane
parking_1
parking_2
first_car_parked == true
first_car_parked = false,
s_len++
first_car_parked == false
first_car_parked = true
park?
unpark?
second_car_parked == false
second_car_parked = true
park?
unpark?
second_car_parked == true
second_car_parked = false,
s_len++
street_beginning
move?
s_len++
find_parking_1
continue
continue_move?
s_len++
first_car_parked == true
s_len++
find_parking_2
continue2
end_of_street?
s_len++
second_car_parked == true
s_len++
end_street
rightLane
parking_1
parking_2
first_car_parked == true
first_car_parked = false,
s_len++
first_car_parked == false
first_car_parked = true
park?
unpark?
second_car_parked == false
second_car_parked = true
park?
unpark?
second_car_parked == true
second_car_parked = false,
s_len++
street_beginning
move?
s_len++
find_parking_1
continue
continue_move?
s_len++
first_car_parked == true
s_len++
find_parking_2
continue2
end_of_street?
s_len++
second_car_parked == true
s_len++
end_street


carOne
carTwo
leftLane
rightLane
start
start
street_beginning
street_beginning
move
parking_spot
find_parking_1
park
start
perform_parking
parking_1
street_beginning
move
parking_spot
find_parking_1
parking_spot
continue
continue_move
parking_spot
perform_parking
parking_1
find_parking_2
unpark
parking_spot
parking_spot
continue
continue_move
parking_spot
parking_spot
find_parking_2
park
perform_parking
parking_2
find_parking_2
perform_parking
parking_2
continue2
unpark
parking_spot
continue2
continue2
end_of_street
parking_spot
exit_street
continue2
end_street
end_of_street
exit_street
exit_street
end_street
end_street

Chapter 3

Related Work

3.1 Properties

1. After observing Encrypted writes,the pages which have encrypted writes should be made invalid and make them available for Garbage collection.

2. Update Physical Location of victim pages in theTime out invalidator queues when their location has changed.

3. EWT provides information of potential victims to TOI, Upon receiving the information check whether it is read by host or not.If read TOI puts a pair of physical location and timestamp into queue(page in the queue is still valid).

4. Page should not be simultaneously in three queues (TPShort,TPLong,Overwritten Queue). LTL : G not(OPQueue and TPShort and TPLong).

5. When a page reaches overwritten state,it always goes to overwritten queue LTL : F(Overwrite $\rightarrow$ G(OPQueue))

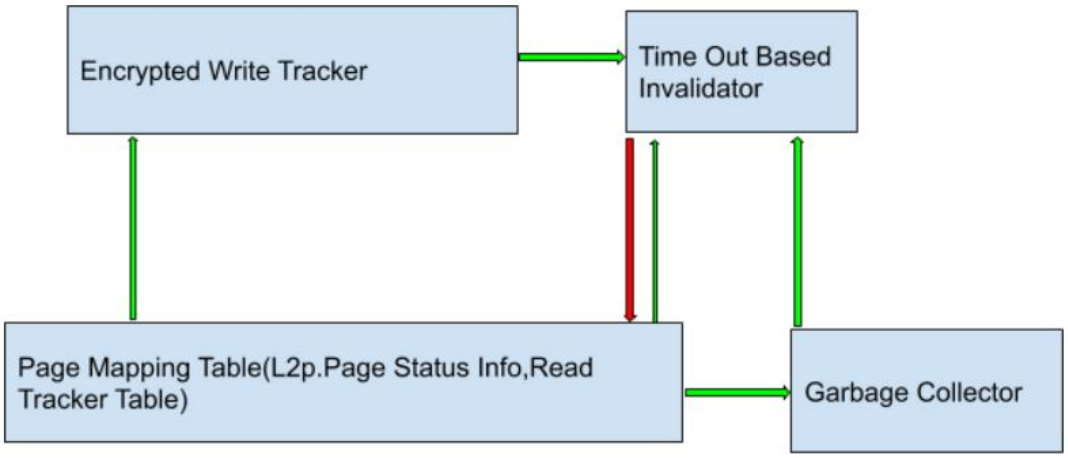
6. When a page reaches trim state,it should always goes to either TP Short Queue or TP Long Queue LTL : F(Trim $\rightarrow$ G(TPShort v TPLong)

7. When a page reaches overwritten state,it always goes to overwritten queue LTL : F(Overwrite $\rightarrow$ G(OPQueue))

8. When the bitmap is zero,it should go to TPShort Queue ,if any encrypted writes occur then page should be sent to TPLong Queue

9. If there are no encrypted writes in the monitoring period the items in the queue should be made available for garbage collection.

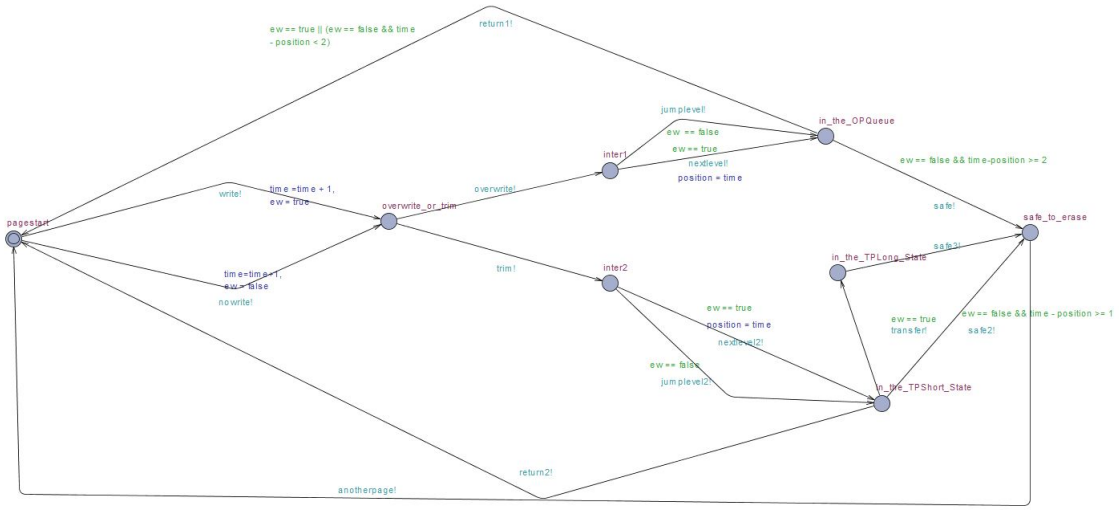
3.1.1 Design of ransom blocker in NAND Flash



Chapter 4

RESULTS

4.1 TOI MODEL



This model contains two templates namely observer template and TOI template .The aim of this model is to insert the page into one of the three TOI Queues namely OPQueue(for Overwritten Pages),TPLong Queue and TP Short Queue.After inserting we will be monitoring the page in the given state with time constraints and check whether the page is safe to erase or it should be monitored for some more time.If encrypted writes are observed during monitoring period ,we will be backing that page for some more time otherwise we will make that page safe to erase state and make available for reclamation.

4.1.1 States

1. Pagestart : starting location of victim page
2. overwrite or trim : based on the action of page(overwrite or trim) path is decided further
3. inter 1 and inter 2 : two intermediate states in the models before going into one of the three queues , the edges from this states helps us to update the monitoring period based on the writes(encrypted or non encrypted)
4. in the OPQueue : the page is entered into the OPQueue state
5. In the TPLong Queue : the page entered into the TPLong queue
6. In the TPShort Queue : the page entered into the TPShort Queue
7. safe to erase : page enters into safe to erase state when it is confirmed that no encrypted writes were observed in the monitoring time.

4.1.2 Edges

Channels present are

1. overwrite,trim : As page should be overwritten or trim
2. nextlevel, nextlevel 2 : one of the edges for entering into respective queues and also update the time value(position variable) with universal clock when encrypted writes observed
3. return1,return2 : various paths to return to the start state so as to monitor the page
4. write,no write : these edges show that whether there is encrypted write or not and monitor the page based on the input
5. jumplevel, jumplevel 2 : one of the edges for entering into respective queues and used when there is no encrypted write while monitoring the page
6. safe,safe2,safe3 : these channels shows that page is safe to make invalid

7. anotherpage:after completion of the page

8. transfer : transfer the page from tp short queue to tp long queue

4.1.3 Variables

int time : global variable for TOI State Machine

Int position : variable for updation of timestamp which is helpful for monitoring (to decide whether it would be safe to erase or return to monitor)

bool ew : used to store status of write for the particular transition

4.1.4 Guards and Updates

1. Write ! Update : time = time + 1 , time incremented ew = true , as write is encrypted

2. Nowrite ! Update : time = time + 1 , time incremented ew = false , as write is non encrypted

3. Jumplevel ! Guard : ew == false , this transition is used as we don't have to update the position(timestamp) of page

4. Nextlevel ! Guard : ew == true , we have to update the position of page Update : position = time (update for monitoring)

5. Return1 and return 2 ! Guard : ew == true or (ew == false and time - position less than k)(need to be monitored for some more time)

6. Transfer ! Guard : ew == true , transfer to TPLong if and only if encrypt write is true

7. Safe ! and safe 2! Guard : ew == false and time - position not less than k(depends on whether overwrite or trim,sample values of 1 and 2 are taken in the above model)

4.1.5 Description of Model and Examples

This TOI State machine is virtually divided into 2 paths,One with overwrite which goes into OPQueue and other with Trim which goes into TPShort and may transfer to TPLong if we observe any encrypted writes in monitoring period for a page in TPShort Queue.In the whole state machine time is global and incremented whether you deal with page1 and page2

4.1.6 path1 execution

Path1 (for overwrite)

Step1)For the page to be inserted(p1) Write → overwrite → inter1 →(nextlevel) OPQueue→ return1

Here return1 is used so that we can monitor this page in the following steps

Step2)We can use this page or we can try to insert another page into one of the three queues

2a) for monitoring the inserted page(p1)

nowrite→ overwrite → inter1 →(jumplevel)OPQueue → return1

2b) can insert the another page(p2)

Write → trim → inter2 →(nextlevel)TPQueue → return2

Step3)We can repeat the above step 2a until we get a safe to erase state

Example Implementation values

Series of steps : Step1 → Step2b → Step 2a → Step 2a

Initially time = 0;

Step1) time = 1,position = 1,return(p1 is inserted)

Step2b) time = 2,position = 2,return(p2 is inserted)

Step2a)time = 3,position = 2,return(does not satisfy guard of safe to erase state and still it should be monitored)

Step2a)time = 4,position = 2,safe(satisfy guard time -position not less than 2) Hence it is safe to erase

4.1.7 path2

Path2(for Trim)

The formula followed is that if any encrypted write is observed while the page is in TPShort it will be pushed to TPLong state

Step1) For the page to be inserted(p2) write→ trim→ inter2 →(nextlevel) → TPShort → return

Step 2a) If encrypted write is observed Write –¿trim → inter2 → (nextlevel) → TPShort →(transfer) TPLong → (safe3)safe to erase

Step 2b)If non encrypted write is observed (i.e monitoring the page in TPShort)

Nowrite → trim → inter2 → (jumplevel) → TPShort →(safe2)safe to erase

Example Implementation values

Ex1 : Step1 → Step 2a Let time = 4

step1)time = 5,position = 5,return

step2a) time = 6,position = 6,transfer from TPShort to TPLong and stays for there for some period of time and then goes to safe to erase state

Ex2 : Step 1 → Step 2b

step 1)time = 5,posiion = 5 return

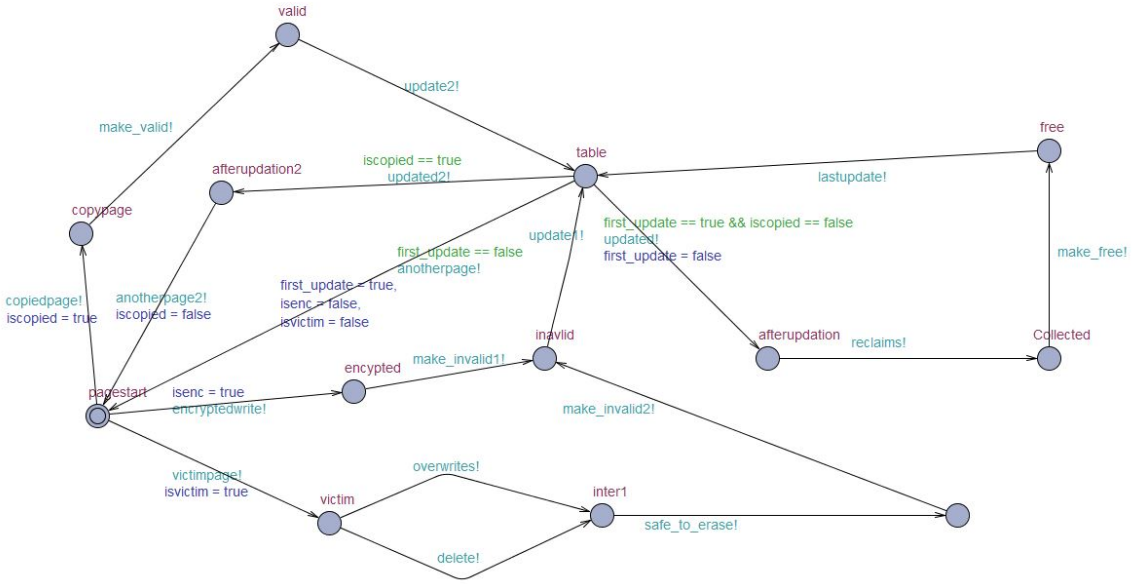
step 2b) time = 6,positon = 5,safe to erase (guard condition satisfy as time - position not less than 1)

)

4.2 Page Status Model

Pagestatus model helps in changing the state of page from one form to another and updates in the table which maintains it's physical locations and page status such as valid,invalid or free.

This model virtually contains three paths namely encrypted write,victim page and copied page.In copied page, while erasing ,when it copies data to other pages in other



block,the status of the page changes from free to valid.If the page is the encrypted write then page status should be made from valid to invalid.If the page is the victim page then after some time it will be make invalid.After going to the invalid state those pages were reclaimed by garbage collector and made free.In this state machine for the paths of encrypted and victim pages two updates are done namely first update and last update ,one for making from valid to invalid and from invalid to free

4.2.1 States

1. Page start : makes transition based on victim,encrypted or copied page(user should know with which page type they are dealing with.
2. encrypted : location where the page is entered into encrypted path
3. copypage : location where the page is entered into copy path
4. victim page : location where the page is entered into victim page path
5. inter1 : intermediate location after the page has overwritten or trimmed in the victim page path
6. safe to erase :it is assumed that all pages which were written or overwritten would likely to be made invalid after achieving safe to erase state

7. invalid,valid : pages were made invalid and valid respectively
8. table : it is a state where updation in the table takes place
9. after updation,after updation 2 : they are the states achieved after updating in the table and proceed further
10. collected : the page collection state after it is made invalid and it is ready for reclamation
11. free : state which represents the free status

4.2.2 Edges with description of guards and invariants

1. copied page channel represents that page transitioned is copied page and Update (iscopied = true) is to maintain the transitions regarding copy
2. encrypted write channel represents that page transitioned is encrypted and Update (isenc = true) is to know that page maintained is encrypted
3. victim channel represents that page maintained is victim and Update (isvictim = true) is to know that page maintained is victim
4. overwrites channel represents that the victim page is overwritten
5. trim channel represents that the victim page is trimmed
6. make invalid1 channel makes the page status of encrypted from valid to invalid
7. make invalid2 channel makes the page status of victim page from valid to invalid
8. safe to erase knows that the overwritten or trim page has monitored for sufficient time and now made safe to erase
9. make valid represents page which is copied is made from free to valid
10. update1 represents update takes place in the table after invalidation
11. update2 represents update takes place in the table after validation

12. updated represents after updation in the table the page is proceeded further(not in the the copy path),Guard (first update == true and iscopied == false),Update (first update == false) are the resticting conditions based on the transitions

13. reclaims represents for reclaiming the page after it was made invalid

14. lastupdate represents update in the table so that it was made from invalid to free

15. updated2 represents after updation in the table the page is proceeded further(in the copy path)and the Guard (iscopied == true) allows transition when iscopied value is true

16. anotherpage and anotherpage2 channel represents the reaching of start state with suitable guards and updates

4.2.3 Implementation examples

1. Pagestart → encrypted → invalid → table → afterupdation → collected → free → table → another page → page start (represent one of the paths of encrypted page)

2. Pagestart → victim → overwrite/trim → inter1 → inter2 → invalid → table → after updation → collected →free → table → page start.(represents one of the path of victim page)

3. Pagestart → copy → valid → table→ after updation → pagestart (represents one of the paths of copy page)

In this Model ,it is assumed that we know whether the page is encrypted write or victim page or copied page in the starting and transitions are based on them.

Chapter 5

Conclusion

We have presented Time out based invalidator model and page status model of ransom blocker which provided a full protection against ransomware attacks.Ransom blocker uses a time out based backup policy which helps SSD not to overly backup as in FlashGuard nor monitor only for shorter period of time as in SSD Insider, RansomBlocker can selectively back up only the data which are highly likely to be attacked by ransomware. As a results, Ransom Blocker showed almost the same performance and lifetime over an SSD without ransomware protection The future development trends of flash memory, such as the widespread adoption of higher-level flash memory and the emerging of three-dimensional (3D) flash memory architectures are being discussed.Integrating RBlocker with AI acceleration chips for more quick and accurate ransomware detection is one of the future plans

References for Research Results

1. Jisung Park ,Youngdon Jung, Jonghoon Won, Minji Kang, Sungjin Lee,Jihong Kim RansomBlocker: a Low-Overhead Ransomware-Proof SSD
2. Johan Bengtsson and Wang Yi , Timed Automata: Semantics, Algorithms and Tools
3. Rajeev Alur and David L. Dill ,A theory of timed automata
4. Gerd Behrmann, Alexandre David, and Kim G. Larsen A Tutorial on Uppaal 4.0
5. M.C.Yang,Y.M.Chang,C.W.Tsao,P.C.Huang Garbage collection and wear leveling for flash memory Past and future
6. Arie Gurfinkel , Model Checking
7. Matheus Pereira Junior,Gliefer Vaz Alves A Study Towards the Application of UPPAAL Model Checker
8. Uppaal (model checking tool) and Correctness Criteria for Beginners
9. Uppaal (model checking tool) and Correctness Criteria for Beginners youtube