A Model Checking Method for Secure Routing Protocols by SPIN with State Space Reduction

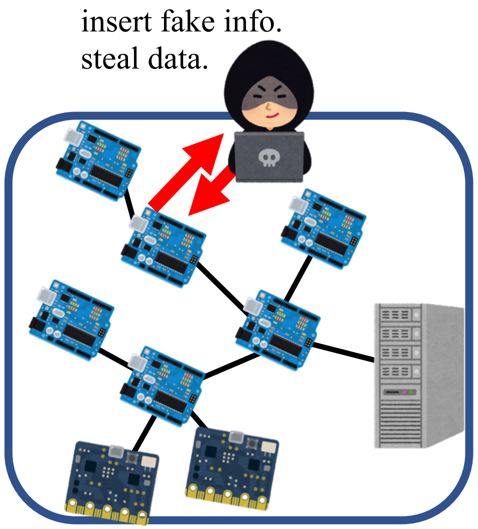
Hideharu Kojima, Naoto Yanai

Graduate School of Information Science and Technology,

Osaka University

Contact:
Hideharu Kojima: hkojima@ist.osaka-u.ac.jp

Introduction



An IoT System on WSN

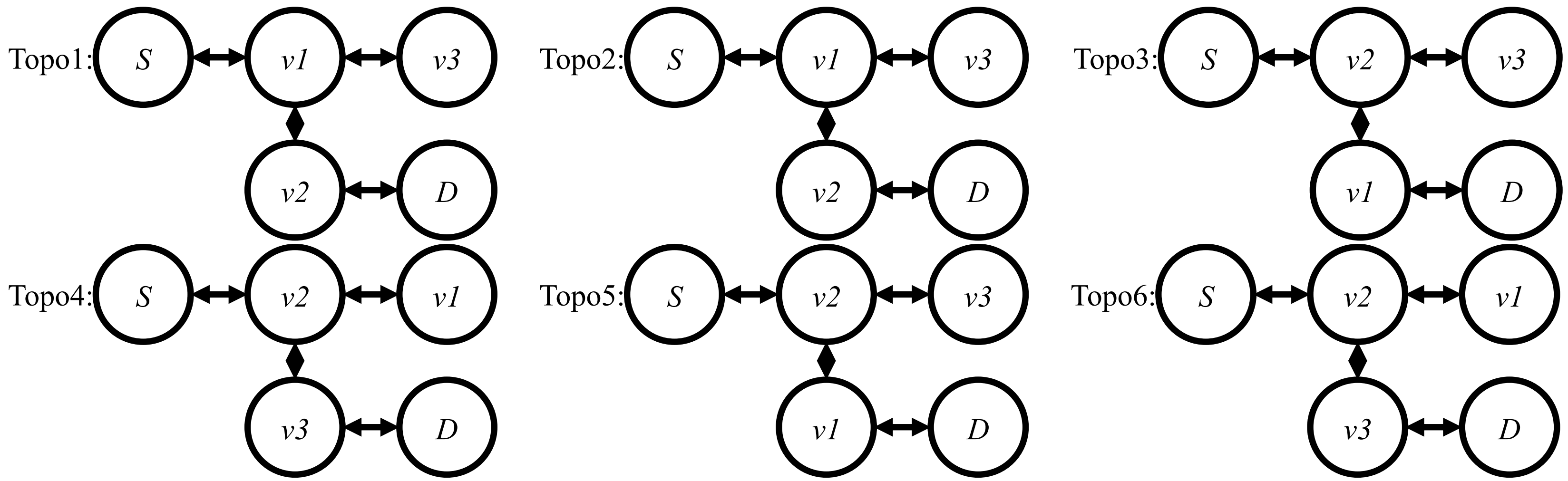
Since WSNs are often essential for developing IoT systems, improving the security for WSNs is an important issue for IoT systems. More specifically, IoT systems are applications running on WSNs to collect information by using sensors efficiently.

It is easy that an attacker in a propagation area of nodes receives and then analyzes packets and send malicious packets. Once malicious attack is succeeded, e.g., in a case of black hole attacks, an attacker can obtain sensing data from sensor nodes instead of reaching the data to sink nodes while commands from the sink nodes are not reached to sensor nodes. Consequently, critical problems will occur in social activities of IoT systems.

To tackle such an issue, secure routing protocols have been investigated. These protocols have introduced digital signatures to guarantee the route information in a packet for route establishment. By applying a secure routing protocol as a routing protocol for WSN, IoT systems is able to sidestep injuries from the attacks.

To apply a secure routing protocol into IoT systems, developers should verify whether the secure routing protocol satisfies the given properties. In this paper, we focus on two kinds of targeting tools for verification of a secure routing protocol, i.e., a formal verification method with the model checking tool named SPIN

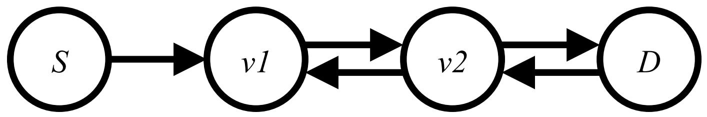
Problem



When model checking is conducted for a target protocol, we need to make a model representing behavior of the target protocol. Then, we can obtain result whether the target protocol satisfies the given properties or not. When the number of nodes is small, e.g., four and five nodes, the model checking can finish quickly. However, executions of the model checking with more than nine nodes is not finished even after the execution time is taken for a day. The reason is because, when the number of nodes becomes large, the number of topologies increases exponentially. We show an example in above figures. The figures shows six topologies consisting of five nodes. There are six orders of the three forwarding nodes, i.e., v1, v2 and v3. These orders of the nodes are permutation consisting of three nodes among the entire nodes. Likewise, other topologies also exist in addition to the topologies in above figures. As the number of topologies is getting huge, the number of states searched by a model checking tool becomes huge exponentially. Consequently, a topology increase is one of concerns for the state space explosion.

Proposed Method

(1)



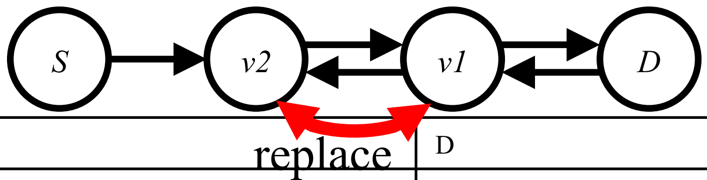
S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,1,2),sig(S,1,2))
	pkts:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))
	c[D]:(0,S,D,0,3,1,ri(S,1,2),sig(S,1,2))

v1	v2

id: v1, PC:3
pktr:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
pkts:(1,D,S,S,3,1,ri(S,1,2,D),sig(S,1,2,D))
c[v1]:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
	id: v2, PC:0
	pktr:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))
	pkts:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
	c[v2]:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))

(2)



S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))
	pkts:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	c[D]:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))

v1	v2

id: v1, PC:0
pktr:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
pkts:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
c[v1]:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	id: v2, PC:3
	pktr:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
	pkts:(1,D,S,S,3,1,ri(S,2,1,D),sig(S,2,1,D))
	c[v2]:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))

replace

The figure (1) and the figure (2) are the same topology. Tables under figures represent state variables of corresponding figures. Both state variables in tables are similar. If state variables of the figure (2) related to the node v1 and the node v2 are replaced, both figures become same state.
The proposed method regard two state which are the same topology as the same sate by replacing forwarding nodes.

The proposed method is based on our previous method [27]. Our previous method suppress the number of states by replacing state variables related to forwarding nodes (target state variables are red colored in the lower table).

However, node information included in signatures (green colored in the lower table) cannot be replaced without recalculation them. The proposed method in this research can replace node information by calculating signatures.

[27]: H. Kojima and N. Yanai, "A state space reduction method for model checking of wireless multi-hop network routing protocols focusing on topologies," in Seventh International Symposium on Computing and Networking Workshops, CANDAR 2019 Workshops, IEEE, 2019, pp. 14–20.

Proposed Method: Node replacement

Replacement Step 1: PC, c, pktr, pkts, ri and sig of v1 in (2) are moved to v2 in s_e. State variables in s_e related to v2 is changed.
Replacement Step 2: PC, c, pktr, pkts, ri and sig of v2 in (2) are moved to v1 in s_e. State variables in s_e related to v1 is also changed.
Replacement Step 3: S and D in (2) are moved to S and D in s_e. State variables in s_e related to S and D are changed.
Replacement Step 4: we replace values ”v1” and ”v2” stored in state variables to ”v2” and ”v1”, respectively. For example, pktr=(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D)) of v1 in Replacement Step 3 becomes pktr=(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,2,1,D)).
Replacement Step 5: signatures in all pkts, pktr and c of all nodes are recalculated. For example, pktr=(1, D, S, v1, 2, 1, ri(S,1,2,D), sig(S,2,1,D)) of v1 in Replacement Step 4 becomes pktr=(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D)).

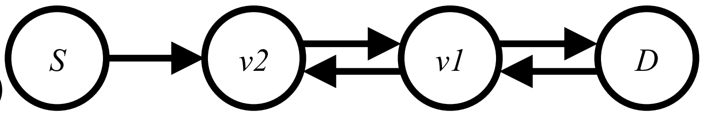
an empty state s_e

S	D
id: S, PC:0	id: D, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())	pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,0,0,0,0,0,ri(),sig())	pkts:(0,0,0,0,0,0,ri(),sig())
c[S]:(0,0,0,0,0,0,ri(),sig())	c[D]:(0,0,0,0,0,0,ri(),sig())
v1	v2
id: v1, PC:0	id: v2, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())	pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,0,0,0,0,0,ri(),sig())	pkts:(0,0,0,0,0,0,ri(),sig())
c[v1]:(0,0,0,0,0,0,ri(),sig())	c[v2]:(0,0,0,0,0,0,ri(),sig())

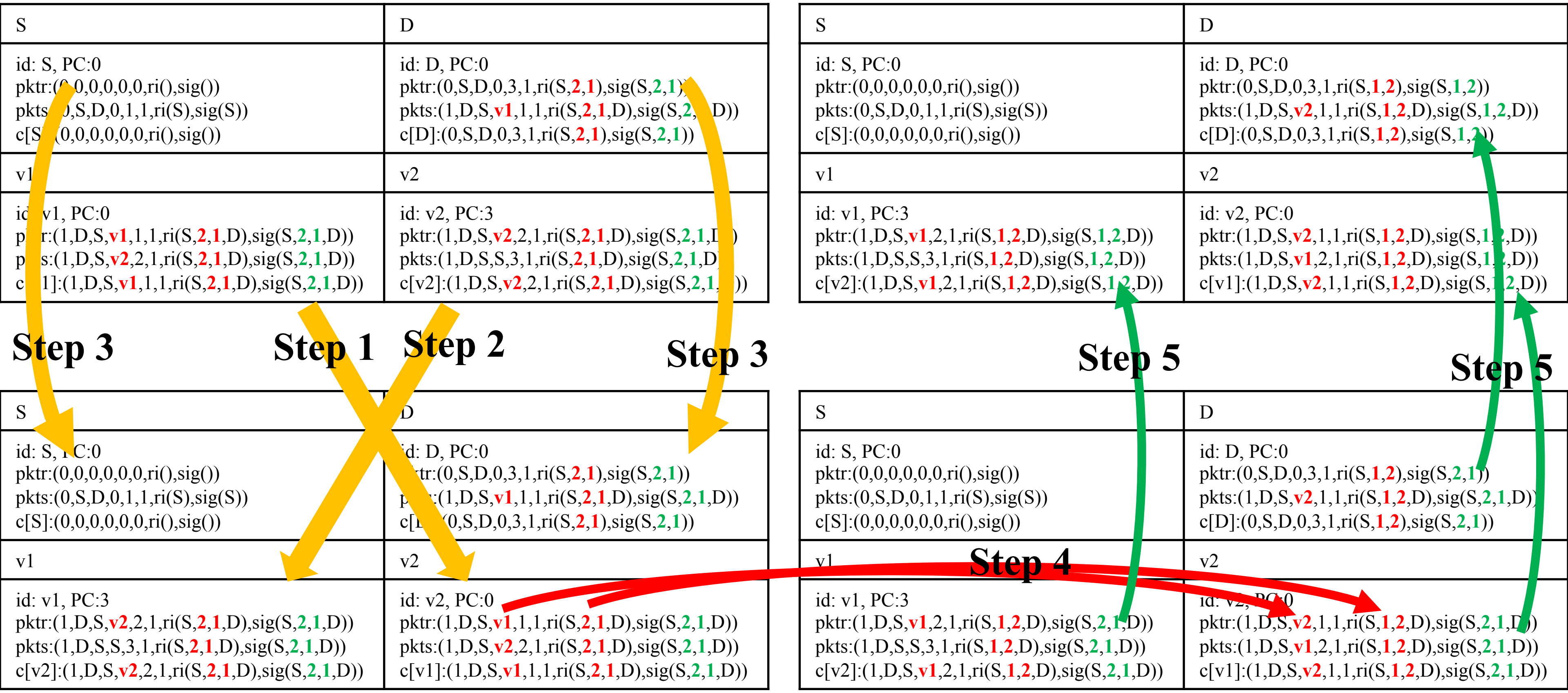
(2) in above phrases is a state representing the figure (2) in the previous slide.
An empty state s_e is prepared for node replacement.
We illustrate node replacement steps in next slide.

Illustration for Node Replacement

state of (2)



Green arrows indicate signature recalculation in **Step 5**.



S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))
	pkts:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	c[D]:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))

v1	**v2**

id: v1, PC:0
pktr:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
pkts:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
c[v1]:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	id: v2, PC:3
	pktr:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
	pkts:(1,D,S,S,3,1,ri(S,2,1,D),sig(S,2,1,D))
	c[v2]:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))

S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,1,2),sig(S,1,2))
	pkts:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))
	c[D]:(0,S,D,0,3,1,ri(S,1,2),sig(S,1,2))

v1	**v2**

id: v1, PC:3
pktr:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
pkts:(1,D,S,S,3,1,ri(S,1,2,D),sig(S,1,2,D))
c[v2]:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
	id: v2, PC:0
	pktr:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))
	pkts:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,1,2,D))
	c[v1]:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,1,2,D))

S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))
	pkts:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	c[D]:(0,S,D,0,3,1,ri(S,2,1),sig(S,2,1))

v1	**v2**

id: v1, PC:3
pktr:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
pkts:(1,D,S,S,3,1,ri(S,2,1,D),sig(S,2,1,D))
c[v2]:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
	id: v2, PC:0
	pktr:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))
	pkts:(1,D,S,v2,2,1,ri(S,2,1,D),sig(S,2,1,D))
	c[v1]:(1,D,S,v1,1,1,ri(S,2,1,D),sig(S,2,1,D))

S	D

id: S, PC:0
pktr:(0,0,0,0,0,0,ri(),sig())
pkts:(0,S,D,0,1,1,ri(S),sig(S))
c[S]:(0,0,0,0,0,0,ri(),sig())
	id: D, PC:0
	pktr:(0,S,D,0,3,1,ri(S,1,2),sig(S,2,1))
	pkts:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,2,1,D))
	c[D]:(0,S,D,0,3,1,ri(S,1,2),sig(S,2,1))

v1	**v2**

id: v1, PC:3
pktr:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,2,1,D))
pkts:(1,D,S,S,3,1,ri(S,1,2,D),sig(S,2,1,D))
c[v2]:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,2,1,D))
	id: v2, PC:0
	pktr:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,2,1,D))
	pkts:(1,D,S,v1,2,1,ri(S,1,2,D),sig(S,2,1,D))
	c[v1]:(1,D,S,v2,1,1,ri(S,1,2,D),sig(S,2,1,D))

Yellow arrows represent processes from **Step 1** to **Step 3**, the proposed method moves state variables from the state (2) to an empty state s_e

Red arrows indicates replacement of state variables "v1" to "v2", "v2" to "v1", and "1" to "2", "2" to "1" in route information *ri* in **Step 4**.

Implementation

We implement the proposed method as C source codes in pan.c generated by SPIN[1] and apply an aggregate signature algorithm proposed in [13] as a signature algorithm in ISDSR. To embed processes of the proposed method, we create a script file, named **isdsr.py** in Python. The processes of **isdsr.py** are followed after the number of nodes N is given. The details are presented as follows:

1) Process 1: **isdsr.py** generates a promela source code, named **isdsr_gen.pml**, which is a model for ISDSR. The generated **isdsr_gen.pml** consists of models containing one source node, one destination node, and N-2 forwarding nodes. In our implementation, a source node has a value 1 as its own id, and a destination node has a value 2 as its own id. Forwarding nodes are assigned more than 3 as their own id.
2) Process 2: The process makes the run of SPIN with **isdsr_gen.pml** in order to generate source codes in the C language, e.g., **pan.c** and **pan.h**.
3) Process 3: The process generates two header files (**replace.h** and **print_state.h**), and two source files **replace.c** and **print_state.c** in C language. The **replace.c** includes functions to replace nodes. The **print_state.c** includes functions to print state variables for debugging.
4) Process 4: The process generates a C language source code, named **pan_symm.c** from **pan.c**. Statements to call functions to replace nodes in **replace.c** are inserted into **pan_symm.c**.

After finishing the above processes, six files, i.e., **aodv_gen.pml**, **pan_symm.c**, **replace.h**, **replace.h**, **print_state.h** and **print_state.c**, are generated. To execute model checking for ISDSR, we also implement **ibsas.c**, **ibsas_for_isdsr.c** and **isdsr_spin.c** for the aggregate signature algorithm in ISDSR.

[1] https://github.com/delab-ou/isdsr spin
[13]: K. Muranaka, N. Yanai, S. Okamura, and T. Fujiwara, "ISDSR: Secure DSR with ID-based Sequential Aggregate Signature," in Proc. the 13th International Joint Conference on e-Business and Telecommunications - Volume 4: SECRYPT, (ICETE 2016), 2016, pp. 376–387.

Experimental Results

No. Nodes	states			time [s]			mem [MB]		
	proposed	original	rate	proposed	original	rate	proposed	original	rate
4	334	542	61.62	0.31	0.22	140.90	131	133	97.76
5	1195	4009	29.78	2.12	0.73	290.41	138	159	86.79
6	6388	29072	21.97	10.3	3.95	260.74	185	387	47.80
7	35284	213419	16.53	108	29.4	367.34	496	2348	21.12
8	191072	1665776	11.47	$1.51x10^3$	225	671.11	2415	19698	12.26
9	1004558	-	-	$2.36x10^4$	-	-	13736	-	-

The experimental results are shown in the table above. We apply a parameter MEMLIM = 60000 as a compile-time option of model checker. This means that memory usage is up to 60 GB because the experimental environment equips 64 GB memory. ”O. M.” in the table represents the out of memory.

This result means that the execution cannot be completed by exhausting the 60 GB memory. The results of the proposed method are superior to those of the original model checking in SPIN. Especially, when the number of nodes is eight, the proposed method reduces the number of states by 11.47% in comparison with the original checking in SPIN. The proposed method is also able to treat states which are the same shape of the topology by replacing nodes. In the experiment with nine nodes, while experiments with the original checking in SPIN do not finish in the setting, those with the proposed method can be finished.

Discussion

In the experimental results, the proposed method takes more calculation time than the original model checking in SPIN. We consider two reasons related to this point. The first reason is that the algorithm to find an equivalent state is not mature. With regard to the execution time, the proposed method needs to take a way to find an equivalent state efficiently.

No. Nodes	**calls**			**Execution time [s]**		
	proposed	original	rate	proposed	original	rate
4	101	10	10.1	$1.42\text{x}10^{-1}$	$1.44\text{x}10^{-2}$	9.90
5	1326	32	41.44	1.91	$4.66\text{x}10^{-2}$	40.97
6	6775	130	52.12	9.69	$1.87\text{x}10^{-1}$	51.76
7	72747	652	111.58	$1.05\text{x}10^{2}$	$1.87\text{x}10^{-1}$	111.26
8	1005265	3914	256.84	$1.43\text{x}10^{3}$	5.69	250.97

The other reason is that the generation of signatures needs a calculation resource. More specifically, the signature algorithm in ISDSR applies an elliptic curve and its related functions for generation and verification of signatures. To recognize influence of the signature generation, we measure the number of calls for signature generation and accumulation of the time for signature generation in one execution.

The columns ”proposed” and ”original” represent the number of calls for signature generation in one execution. The column of “ratio” represents values from the expression ***proposed/original***. The number of calls in the column “proposed” is much greater than that of “original”. In the row whose number of nodes is seven, the value of “proposed” is as over one hundred times as the value of “original”. We focus on the ratio how the execution time for signing function accounting for in the execution of model checking. For example, when the number of nodes is eight, the execution of the proposed method takes $1.51\text{x}10^{3}$ seconds in the table of the experimental results. This execution time includes $1.43\text{x }10^{2}$ seconds for signing function. The ratio of the execution time for signing function accounts for about 94% for the execution of model checking. On the other hand, the ratio of the execution time for signing function in original accounts for about 2%. The execution times of the proposed and the original except the signing function execution are about 90 seconds and about 220 seconds, respectively. As long as the execution time except the signing function execution, we consider the proposed method takes less time for model checking.

Conclusion

In this paper, we proposed a method to reduce the state space for verification of ISDSR. The proposed method has been based on symmetry reduction and the main idea is to focus on shapes of topologies. We improved the proposed method to treat signatures in state variables for secure routing protocols. If shapes of both topologies represented by different states are identical, the proposed method can check if the different states are equivalent by replacing nodes and calculating signatures. We also implemented the proposed method and conducted experiments to measure the number of states, memory usage and execution time in comparison with the original checking in SPIN. As a result, the proposed method outperformed the original checking in SPIN. Furthermore, the proposed method completed an execution for nine nodes while the original checking in SPIN was unable to finish the execution of the same conditions due to the memory limit.

The current limitation in this work is that the proposed method is specialized for ISDSR. Hence, in future work, we plan to apply the proposed method to verification of other secure routing protocols.
We also try to introduce the idea of the proposed method into SPIN itself.
Another future plan is to implement an efficient method for finding an equivalent state to overcome the bottleneck described in Discussion.