

Acceleration of a Client Based DNSSEC Validation System in Parallel with Two Full-Service Resolvers

1st Yong Jin^{*†}, 2nd Kazuya Iguchi^{†§}, 3rd Nariyoshi Yamai^{†¶}, and 4th Masahiko Tomoishi^{*||}

^{*}Tokyo Institute of Technology, 2-12-1 O-okayama, Meguroku, Tokyo, JAPAN 152-8550

[†]Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho, Koganei, Tokyo, JAPAN 184-8588

[‡]yongj@gsic.titech.ac.jp, [§]kazuigu@net.cs.tuat.ac.jp, [¶]nyamai@cc.tuat.ac.jp, ^{||}tomoishi@noc.titech.ac.jp

Abstract—Domain Name System Security Extensions (DNSSEC) is an effective solution for mitigating DNS cache poisoning attacks but it also has many issues on real operations. In the literature, a client based DNSSEC validation system has been proposed in order to solve some existing issues by running DNSSEC validation on each end client. However, in the system, the DNS responses with successful validation will be only used by each end client thus without sharing it with other end clients. In this paper, we propose an acceleration method for the client based DNSSEC validation system by using two different types of external DNS full-service resolvers (DNSSEC-enabled and DNSSEC-disabled) in parallel. As a result, the performance of domain name resolution can be improved by sharing the DNS responses with successful DNSSEC validation. Furthermore, when the domain name resolution on the DNSSEC-enabled external DNS full-service resolver ends with timeout or failure, the end client conducts the DNSSEC validation by itself by querying necessary DNS records to the DNSSEC-disabled external DNS full-service resolver and also a pop-up alert message will warn the users if the DNSSEC validation fails. We implemented a prototype system and evaluated the features as well as the performance of the proposed method. The results confirmed that the proposed method achieved high performance in domain name resolution.

Index Terms—DNS, DNSSEC, client based DNSSEC validation, full-service resolver, DNS cache poisoning attack

I. INTRODUCTION

Domain Name System (DNS) [1] has been widely deployed in the Internet nowadays and DNS based domain name resolution service plays an indispensable role. Since DNS based domain name resolution service is one of the open Internet services, the attacking surface is public to all the Internet users including the cyber attackers. One of the famous attacks against DNS is cache poisoning attack [2] and the number of victims is still on increasing. DNS cache poisoning attack attempts to injects fake DNS information such as fake Name Server (NS) and Address (A) records into the DNS full-service resolvers in order to make the end clients access to the malicious web sites. Consequently, the end clients can be infected by some type of malware and furthermore the privacy information and confidential data can be breached.

As a solution to prevent the DNS cache poisoning attacks, Domain Name System Security Extensions (DNSSEC) [3] has been standardized. By enabling DNSSEC validation on DNS full-service resolvers, the DNS cache poisoning attacks can be effectively prevented. However, DNSSEC causes several issues such as increasing workload on DNS full-service resolvers and

non-integrity provided on the links between the end clients and the DNS full-service resolvers. Furthermore, in case of DNSSEC validation failure, the DNS full-service resolvers just reply “SERVFAIL” result even if it is caused by misconfiguration. In order to solve these issues, in the literature, we have proposed a client based DNSSEC validation system [5]. In this system, some side affects caused by DNSSEC had been solved but the inefficiency problem remained due to the low cache hit rate. Since the DNSSEC validation will be performed on each end client and the validated DNS responses cannot be shared with other end clients.

In this paper, we propose an acceleration method for the client based DNSSEC validation system by using two different types of external DNS full-service resolvers in parallel in order to share the domain name resolution results with successful DNSSEC validation. As a result, even if the end clients have no cached DNS resource records with successful DNSSEC validation, the end clients are also able to obtain them from the DNS full-service resolver with DNSSEC enabled.

II. OVERVIEW OF DNS/DNSSEC AND RELATED WORK

A. Overview of DNS and DNSSEC

DNS translates domain names to the corresponding information such as IP address (A record) and Mail eXchange server (MX record), etc. In general, each end client sends recursive DNS query to the DNS full-service resolver and the DNS full-service resolver performs the domain name resolution instead. The DNS full-service resolver is also called as DNS cache server since it caches all received results. Accordingly, when the end clients query the same DNS records cached in the DNS full-service resolver, the answers will be replied directly from the cache by the DNS full-service resolver.

Since the DNS full-service resolvers only have caches received from the DNS authoritative servers, the attackers can conduct DNS cache poisoning attacks and injects fake DNS resource records into the DNS full-service resolvers. As a solution for mitigating the risk of DNS cache poisoning attacks, source port randomization [4] on DNS full-service resolvers has been proposed and standardized which can barely decrease the success possibility. The fundamental solution is DNSSEC which is mainly deployed on DNS full-service resolvers. DNSSEC uses public cryptography based digital signature and validation for the integrity of DNS responses received from the DNS authoritative servers.

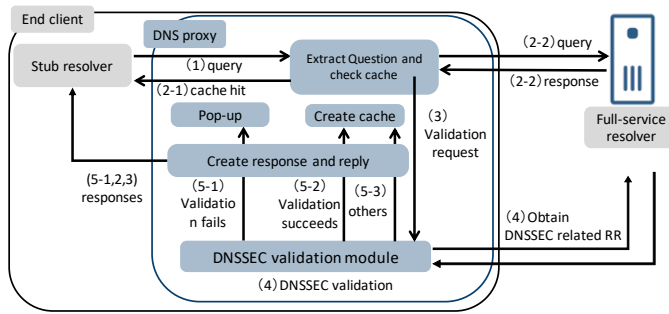


Fig. 1. Domain name resolution with client based DNSSEC validation.

B. Related work

We mentioned that enabling DNSSEC on DNS full-service resolvers may cause several issues. In this section, we introduce some related work with focusing on the following issues.

- 1) The workload of DNS full-service resolvers increases due to the heavy DNSSEC validation overhead.
- 2) When DNSSEC validation fails, a DNS full-service resolver only replies "SERVFAIL" without any DNS resource records which ends with name resolution failure.
- 3) From a DNSSEC-disabled DNS full-service resolver, the end clients cannot obtain DNSSEC validated responses.
- 4) DNSSEC provides integrity for DNS responses between DNS authoritative servers and full-service resolvers but it cannot cover the link to the end clients.

In order to solve the above issues, we have proposed a client based DNSSEC validation system [5] in the literature. Figure 1 presents the workflow of domain name resolution on the system. For the sake of simplicity, we used an explicit DNS full-service resolver as the query destination of the system.

- 1) Step 1: When the stub resolver receives a DNS query, it sends the query to the DNS proxy of the client based DNSSEC validation system.
- 2) Step 2: The client based DNSSEC validation system analyzes the received DNS query packet.
 - 2-1: If the end client has the answer in its cache, the system replies it back to the stub resolver directly.
 - 2-2: Otherwise, the system sends the DNS query to the external DNS full-service resolver with CD flag (DNSSEC Checking Disabled) and goes to step 3.
- 3) Step 3: The system forwards the query name, type, class and flags to the DNSSEC validation module.
- 4) Step 4: The DNSSEC module achieves necessary DNS resource records from the external full-service resolver and performs DNSSEC validation.
- 5) Step 5: If the DNSSEC validation finishes in time (before timeout), the system replies the answer back to the stub resolver based on the following conditions. Otherwise, the system replies the DNS response achieved on step 2-2 back to the stub resolver and pop-up a warning message as illustrated on Fig. 3.
 - 5-1: If the DNSSEC validation fails, the system replies the achieved DNS response back to the stub re-

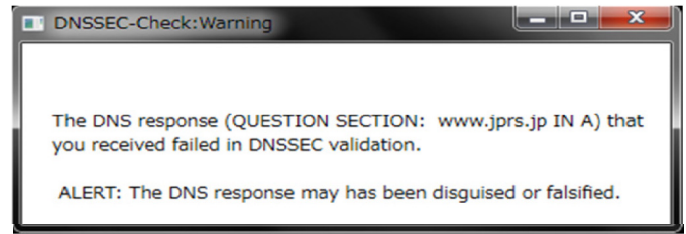


Fig. 2. Domain name resolution with client based DNSSEC validation.

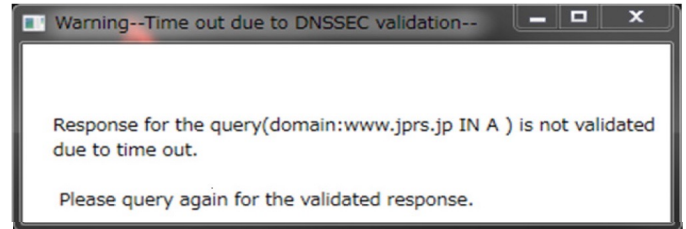


Fig. 3. Domain name resolution with client based DNSSEC validation.

solver. Then the system displays a pop-up message warning the user about the risk as shown in Fig. 2.

- 5-2: If the DNSSEC validation succeeds, the system creates the corresponding DNS response and replies it back to the stub resolver. Then the DNS response will be cached on the end client.
- 5-3: If the query name or type does not support DNSSEC on the authoritative server, the system just replies the received DNS response to the stub resolver and also caches it on the end client.

With the above procedure, in the previous work, the DNSSEC validation will be performed on the end clients thus the issue of increasing workload of the DNS full-service resolvers can be solved. The system also has cache feature on the end clients that the obtained DNS answers with successful DNSSEC validation will be cached [6]. As a result, when the system received the same query, the answer will be replied from the cache without performing DNSSEC validation again.

III. SHARING OF THE DNS RESOURCE RECORDS WITH SUCCESSFUL DNSSEC VALIDATION

According to the measurement results of the previous work, the domain name resolution with client based DNSSEC validation takes about 100 - 200 milliseconds and it can increase to more than 1000 milliseconds depending on the computing capacity of the end clients. On the contrary, by using the cache feature, the same domain name resolution requests from the second time only costs about 20-40 milliseconds which means that the sharing of DNS resource records with successful DNSSEC validation among the end clients can improve the performance. In order to solve this problem, we considered several solutions about how to share DNS resource records successfully DNSSEC validated between multiple end clients.

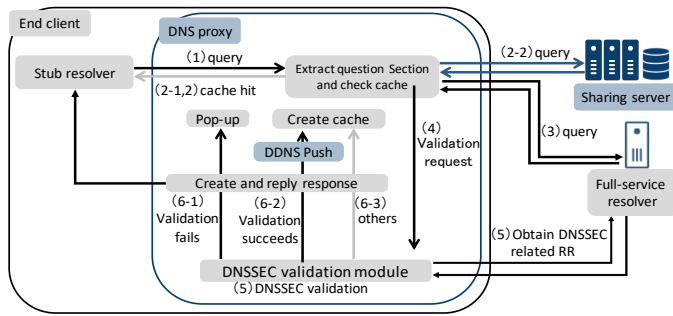


Fig. 4. A client based DNSSEC validation system with using a sharing server.

A. Use a sharing server

The first solution we considered is that to use a sharing server for DNS resource records with successful DNSSEC validation. Each end client updates the sharing server using dynamic DNS update once it obtains a DNS response with successful DNSSEC validation and each end client queries the sharing server first in prior to querying the external DNS full-service resolver for domain name resolution. Therefore, the sharing server works as a non-recursive DNS resolver and each end client achieves the TSIG (Transaction Signature) [7] key from the sharing server in order to perform the DNS update.

Fig. 4 shows the flow of domain name resolution with using a sharing server. The whole flow is similar to that described in section II-B and the difference in here is that each end client queries the sharing server first before querying the external DNS full-service resolver. If the end client obtains the DNS answer from the sharing server, it will be replied back to the stub resolver. Otherwise, the end client queries the external DNS full-service resolver and conducts DNSSEC validation by itself. However, there are also some issues in this solution.

- 1) Malware infected end clients may intendedly add fake DNS resource records to the sharing server.
- 2) Since the sharing server works as an authoritative DNS server so that the TTL (Time To Live) of each DNS resource record will not be decreased automatically.
- 3) The management for TSIG key including creation, allocation and update need high administration cost.

B. Use a DNSSEC-enabled sharing server

In order to solve the issues in the solution of using a sharing server, we considered to enable DNSSEC validation in the sharing server. Specifically, a new process will be launched in the sharing server and once a DNS resource record updated by an end client the process performs DNSSEC validation for it. However, we found some other issues at the design stage.

- 1) It takes time to perform DNSSEC validation for the updated DNS resource records. In addition, the sharing server cannot distinguish if a DNS resource record have been finished with DNSSEC validation or not.
- 2) In the solution of using the sharing server, the sharing server does not need to access to the Internet since it just needs to check if the queried DNS resource

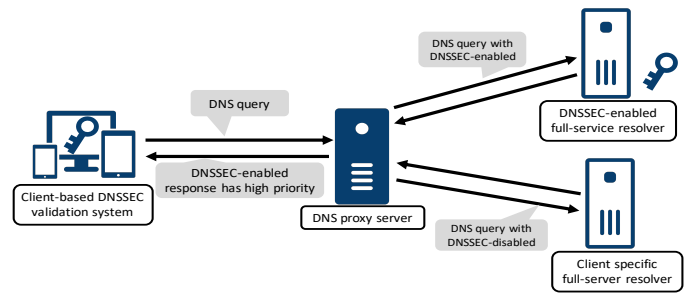


Fig. 5. DNS proxy server queries two DNS full-service resolvers in parallel

records are in the authoritative DNS server. However, if the DNSSEC validation feature is enabled, the sharing server needs to access to the Internet and it increases the risk of being attacked.

C. Use a DNS proxy with two DNS full-service resolvers

As the third solution, we considered to use a DNS proxy server with involving two DNS full-service resolvers on each end client, one is recursive and DNSSEC-enabled (hereafter DNSSEC-enabled DNS full-service resolver) and another is to be used by the client based DNSSEC validation system only (hereafter DNSSEC-disabled DNS full-service resolver). The system topology is shown in Fig. 5. When an end client sends a DNS query to the DNS proxy server, the DNS proxy server queries both of the two DNS full-service resolvers in parallel. If the DNS proxy server receives DNS responses from both of the two DNS full-service resolvers, it replies that from the DNSSEC-enabled DNS full-service resolver with high priority. The client based DNSSEC validation system finishes the domain name resolution when it receives the DNS response from the DNSSEC-enabled DNS full-service resolver first, otherwise the system performs DNSSEC validation by itself. The DNS response can be identified based on the CD (Checking Disabled) and AD (Authenticated Data) flags on the DNS message header. However, this solution may cause the following issues.

- 1) In most cases, the DNS proxy server receives the DNS responses at approximately the same time from both of the two DNS full-service resolvers for a DNS query. Therefore, it needs to add some delay on the response from the DNSSEC-disabled DNS full-service resolver in order to use the DNS response replied by the DNSSEC-enabled DNS full-service resolver. It means that if the DNSSEC-disabled DNS full-service resolver has no cache for a DNS query, the response will be late or be "SERVFAIL" so that the extra delay will not be necessary. In addition, adding delay generates overhead for replying back the response to the end client.
- 2) The CD and AD flags in the DNS response were used to distinguish the DNS responses from DNSSEC-disabled and DNSSEC-enabled DNS full-service resolvers. Therefore, it may not work properly if the DNS proxy server program changes.

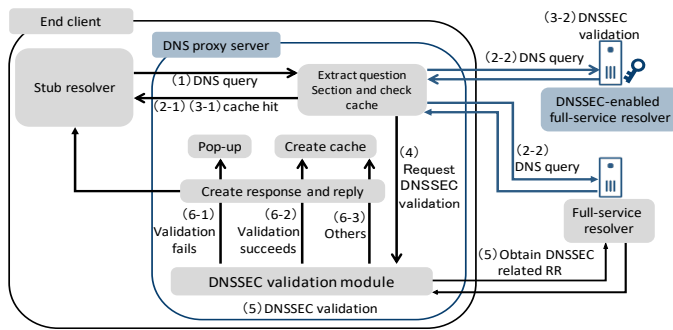


Fig. 6. Name resolution flow with two DNS full-service resolvers in parallel.

IV. ACCELERATION SOLUTION BY USING TWO DNS FULL-SERVICE RESOLVERS IN PARALLEL

In the solution of using a DNS proxy server involving two DNS full-service resolvers, as being mentioned in section III-C, the DNS proxy server had to wait for the DNS response from the DNSSEC-enabled DNS full-service resolver. In order to solve the issue, we propose an acceleration solution in which the client-based DNSSEC validation system directly uses the two DNS full-service resolvers. Specifically, the stub resolver sends all DNS queries to the client-based DNSSEC validation system and the system forwards the queries to the two DNS full-service resolvers in parallel. Then the DNS response replied first will be used by the applications while the later one will be dropped.

For the first time of DNS query on each end client, the DNS response from the DNSSEC-enabled DNS full-service resolver will be slower than the one from the DNSSEC-disabled DNS full-service resolver due to the overhead of DNSSEC validation. However, from the second time for the same query, the DNS response from the DNSSEC-enabled DNS full-service resolver will arrive first. Furthermore, in case of a timeout occurs on one of them, the DNS response from the other DNS full-service resolver can be used. As a result, in the proposed solution, not only the acceleration of the domain name resolution but also the redundancy can be provided. The flow of the domain name resolution in the proposed solution is described in the following with using Fig. 6.

- 1) The stub resolver sends DNS query to the client-based DNSSEC validation system (DNS proxy server).
- 2) The client-based DNSSEC validation system analyzes the DNS query packet.
 - 2-1 If the answer is in the cache, it will be replied to the stub resolver and the name resolution finishes.
 - 2-2 Otherwise, the client-based DNSSEC validation system queries the two DNS full-service resolvers in parallel. In this step, only the query to the DNSSEC-disabled DNS full-service resolver will be set CD flag with 1. Then goes to step 3.
- 3) The client-based DNSSEC validation system receives DNS responses from the two DNS full-service resolvers.

3-1 If the DNS response from the DNSSEC-enabled DNS full-service resolver arrives first, the DNS response will be replied to the stub resolver. Then the system goes to step 4 to perform the client-based DNSSEC validation in order to cache the DNS response after successful DNSSEC validation.

3-2 If the system receives the DNS response from the DNSSEC-disabled full-service resolver first, the system goes to step 4 to perform client-based DNSSEC validation. The DNS query to the DNSSEC-enabled full-service resolver finishes.

- 4) The system forwards DNS query name, type and class to the DNSSEC validation module.
- 5) The DNSSEC validation module queries DNSSEC related resource records to the DNSSEC-disabled DNS full-service resolver and starts DNSSEC validation.
- 6) If the DNSSEC validation finishes in time (before without) then the DNS response will be replied to the stub resolver based on the following conditions. Otherwise, the system replies the DNS response received at step 2-2 to the stub resolver and displays a pop-up alert as shown in Fig. 3. It should be noted that if the system receives the DNS response from the DNSSEC-enabled DNS full-service resolver before the client-based DNSSEC validation finishes, the response will be used instead.
 - 6-1 If the DNSSEC validation fails, the answer will be replied to the stub resolver as is. Similarly, a pop-up alert as shown in Fig. 2 will be displayed to warn the user about the risk of using the answer.
 - 6-2 If the DNSSEC validation succeeds, the system creates and replies the answer to the stub resolver and also caches it in the system.
 - 6-3 If the queried name and type do not support DNSSEC, the system replies the answer to the stub resolver and caches it in the system.

V. IMPLEMENTATION AND EVALUATION

A. Prototype implementation

A prototype system of the proposed solution was implemented by adding new features to the client-based DNSSEC validation system described in section II-B. The DNS proxy server was implemented by using the Perl module named Net::DNSServer::Proxy [8] and the DNSSEC validation module was implemented by using Net::DNS::SEC::Validator [9]. We customized Net::DNSServer module [10] in order to control the DNS proxy server in the prototype system. Figure 7 presents the flow of the proposed acceleration solution of the client-based DNSSEC validation system. The procedure is similar to the one described in section IV. In the flow, the “\$status” indicates the result of DNSSEC validation and the values 1, 3, and 4 mean failure while 128, 133 and 134 mean success. Other status includes non-support query name and type for DNSSEC or unfinished DNSSEC validation.

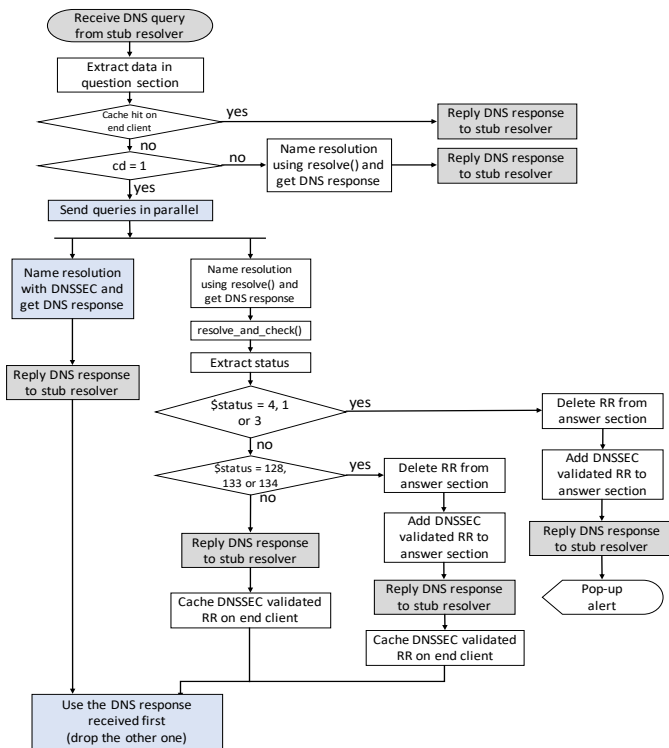


Fig. 7. Flow of name resolution with two full-service resolvers in parallel.

In the prototype system, we considered two types of parallel query method. The first method is to use “Fork” feature to realize parallelization in which one process performs client based DNSSEC validation on the end client and the other process performs domain name resolution by using the DNSSEC-enabled DNS full-service resolver. The second method is that the DNS query will be sent to the DNSSEC-enabled DNS full-service resolver and a specific process for replying the DNS response back to the stub resolver will be launched (hereafter, we call the process as the name resolution specific process). The client based DNSSEC validation system sends the DNS response packet to the name resolution specific process using inter-process communication before starting DNSSEC validation and the name resolution specific process replies the DNS response back to the stub resolver independently. A perl module named IO::Socket::INET [11] was used for the inter-process communication in the prototype system. Finally, we used Berkeley Internet Name Domain (BIND) [12] as the DNS server program in the two external full-service resolvers.

B. Performance evaluation

We conducted performance evaluation on the prototype system using four domain names in a local experimental network. We measured the query time using “dig” command by querying the “A” record of the domain names on several patterns. The four domain names were “google.com.” (non-DNSSEC), “jprs.jp.” and “www.jprs.jp.” (DNSSEC) and “ietf.org.” (DNSSEC and out-of-bailiwick domain name on its authoritative DNS server). We choose three different types

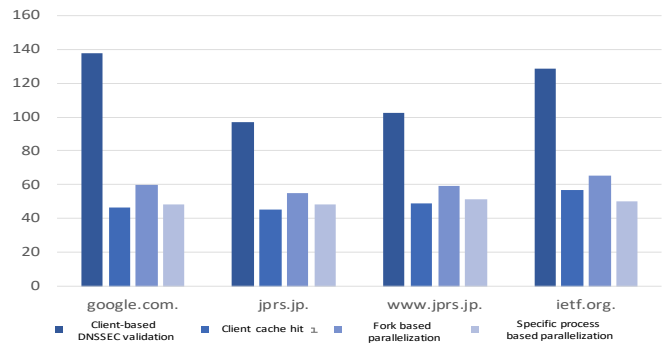


Fig. 8. Results measured on the end client 1.

TABLE I
QUERY TIME OBTAINED ON THE END CLIENT 1 (MILLISECOND)

	<i>Client based DNSSEC validation</i>	<i>Client cache hit</i>	<i>Fork parallel</i>	<i>Special process</i>
google.com.	137.6	46.4	60.2	48.6
jprs.jp.	97.0	45.6	54.8	48.6
www.jprs.jp.	102.4	48.8	59.2	51.6
ietf.org.	128.8	56.8	65.6	50.4

of DNSSEC-enabled domain names in order to compare the DNSSEC validation time in different domain name hierarchies.

Two end clients were used in the performance evaluation. The results measured on the end client 1 are shown in Fig. 8 and Table I. We can see that the client based DNSSEC validation cost about 100 - 150 milliseconds and it decreased to about 40 milliseconds from the second time of the same query (“client cache hit” column). The right two columns of Table I shows the results of the cases that client based DNSSEC validation with cache hit and the use of DNSSEC-enabled DNS full-service resolver with cache hit.

Next, the results measured on the end client 2 are shown in Fig. 9 and Table II. From the measured results, we can see that since the end client 1 had finished the queries for the four domain names already thus when the end client 2 sent the same queries it could obtain the results from the cache within about 20 - 30 milliseconds even from the first time query.

VI. CONCLUSIONS

In this paper, we proposed an acceleration solution for a client based DNSSEC validation system by using two DNS full-service resolvers (DNSSEC-enabled and DNSSEC-disabled DNS full-service resolvers) in parallel in order to reduce DNS query latency and improve domain name resolution performance. The users can conduct secure and fast domain name resolution by using the client based DNSSEC validation system with the proposed acceleration method. Moreover, the proposed acceleration method also provides redundancy by using two DNS full-service resolvers in parallel.

For the future work, we plan to add additional features including flexible process in the parallelization using the end client specific process in order to decrease the DNSSEC validation delay on the client based DNSSEC validation system.

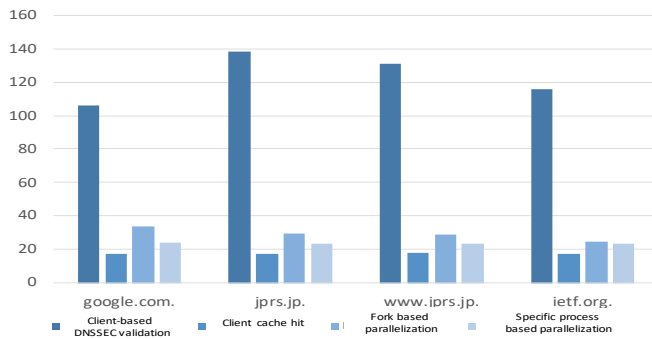


Fig. 9. Result measured on the end client 2.

TABLE II
QUERY TIME OBTAINED ON THE END CLIENT 2 (MILLISECOND)

	<i>Client based DNSSEC validation</i>	<i>Client cache hit</i>	<i>Fork parallel</i>	<i>Special process</i>
google.com.	106.0	17.6	33.6	24.4
jprs.jp.	138.0	17.2	29.4	23.2
www.jprs.jp.	131.2	18.0	28.8	23.4
ietf.org.	115.8	17.4	25.0	23.6

Consequently, when the DNS response from the DNSSEC-enabled DNS full-service resolver arrives first, it will be replied back to the stub resolver and the DNSSEC validation process on the end client will be stopped. Furthermore, we also plan to deploy the proposed system in a real network environment and provide secure and effective domain name resolution service for the internal computers.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI (Grants-in-Aid for Scientific Research) Grant Number 19K20254, 18K11291.

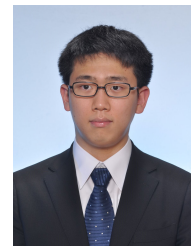
REFERENCES

- [1] P. Mockapetris, "Domain names - implementation and specification," RFC 1035, RFC Editor, November 1987.
- [2] Kaspersky, "What is DNS Cache Poisoning and DNS Spoofing?" (online), available from <https://usa.kaspersky.com/resource-center/definitions/dns> (accessed 2021-06-05).
- [3] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, "DNS Security Introduction and Requirements", IETF RFC4033, March 2005.
- [4] A. Hubert and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers", IETF RFC5452, January 2009.
- [5] Y. Jin, K. Kakoi, N. Yamai, N. Kitagawa, and M. Tomoishi, "A Client Based DNSSEC Validation System with Adaptive Alert Mechanism Considering Minimal Client Timeout," IEICE Transactions on Information and Systems, Vol.E100-D, No.8, pp.1751-1761, Aug. 2017.
- [6] K. Kakoi, Y. Jin, N. Yamai, N. Kitagawa and M. Tomoishi, "Cache Function Activation on a Client Based DNSSEC Validation and Alert System by Multithreading," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017, pp. 37-42, doi: 10.1109/COMPSAC.2017.78.
- [7] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)," IETF RFC2845, May 2000.
- [8] Brown, R.: Net::DNSServer::Proxy - Forwards requests to another DNS server (online), available from <https://metacpan.org/pod/Net::DNSServer::Proxy> (accessed 2021-06-05).

- [9] Marzot, G. S.: Net::DNS::SEC::Validator (online), available from <https://metacpan.org/pod/Net::DNS::SEC::Validator> (accessed 2021-06-05).
- [10] Brown, R.: Net::DNSServer - Perl module to be used as a name server (online), available from <https://metacpan.org/pod/Net::DNSServer> (accessed 2021-06-05).
- [11] Rinaldo, T.: IO::Socket::INET - Object interface of AF_INET domain sockets (online), available from <https://metacpan.org/pod/IO::Socket::INET> (accessed 2021-06-05).
- [12] Internet Systems Consortium, "BIND 9" (online), available from <https://www.isc.org/downloads/bind/> (accessed 2021-10-25).



Yong Jin received M.E. degree in electronic and information systems engineering and Ph.D. degree in Industrial Innovation Sciences from Okayama University, Japan in 2009 and 2012, respectively. In April 2012, he joined the Network Architecture Laboratory of National Institute of Information and Communications Technology, Japan, as a researcher. From October 2013, he joined the Global Scientific Information and Computing Center of Tokyo Institute of Technology. His research interests include network architecture, network security, traffic engineering and Internet technology. He is a member of IPSJ and IEEE.



Kazuya Iguchi received B.E. and M.E. degrees from Tokyo University of Agriculture and Technology, Japan in 2019 and 2021 respectively. His research interests include computer network and Internet Technology.



and IPSJ.

Nariyoshi Yamai received the B.E. and M.E. degrees in electronic engineering and the Ph.D. degree in information and computer science from Osaka University in 1984, 1986, and 1993, respectively. After working at Nara National College of Technology, Osaka University, and Okayama University, he has been a Professor at the Institute of Engineering, Tokyo University of Agriculture and Technology since April 2014. His research interests include distributed systems, network architecture, and Internet. He is a member of IEEE CS, IEEE ComSoc, IEICE



verification and network/security management. He is a member of IEEE.

Masahiko Tomoishi received the B.E. and M.E. degrees in Computer Science from Tokyo Institute of Technology in 1989 and 1991, respectively. In 1994, he joined the Department of Computer Science, Tokyo Institute of Technology, as a research associate. In 2007, he joined National Institute of Informatics, as a specially appointed research associate. In 2008, he joined Tokyo Institute of Technology, as a specially appointed assistant professor. Since April 2011, he has been a professor in Tokyo Institute of Technology. His research interests include software