Verification of DNSsec Delegation Signatures

Florian Kammüller
Middlesex University London
f.kammueller@mdx.ac.uk

***Abstract*—In this paper, we present a formal model for the verification of the DNSsec Protocol in the interactive theorem prover Isabelle/HOL. Relying on the inductive approach to security protocol verification, this formal analysis provides a more expressive representation than the widely accepted model checking analysis. Our mechanized model allows to represent the protocol, all its possible traces and the attacker and his knowledge. The fine grained model allows to show origin authentication, and replay attack prevention. Most prominently, we succeed in expressing Delegation Signatures and proving their authenticity formally.**

***Index Terms*—DNSsec, Isabelle/HOL, authentication, chain of trust, delegation signatures.**

I. Introduction

The Domain Name System (DNS) does not use strong security measures and is thus vulnerable for false IP addresses faked by an attacker – so-called spoofing. The attacker can exploit this weakness and manipulate clients to connect not to the intended server but instead to another possibly malicious one [3]. Domain Name System Security Extensions (DNSsec) [2] is a more recent update of the original DNS protocol to address this security issue. It employs public key cryptography: domain name servers use signatures to authenticate the resolutions of alphanumeric names to numeric values. The additional complexity of DNSsec may lead to misconfigurations and new attacks. Formal models of the DNSsec protocol have been rigorously analysed using model checking tools [4]. However, none of the formal approaches to verifying the DNSsec protocol extension uses the expressive and powerful technique of Paulson's inductive approach [12] to security protocol verification.

In this paper, we use this approach to provide a formal machine checked proof of one of the most important features of DNSsec, the Delegation Signatures (DS). In order to make origin authentication work, it is crucial that the used public keys are themselves certified to be trustworthy using a chain of trust leading to the trust anchor, the root key. We prove the security of this chain of trust in Isabelle/HOL thus providing logical machine checked verification.

In this paper, we first review the prerequisites for our work: the DNSsec protocol and the inductive approach to security protocol verification with Isabelle/HOL in Section II. The model and analysis of DNSsec is based on the adaptation of the inductive approach in Isabelle/HOL to IP-protocols [8]. Besides being capable of scaling up our approach, using Isabelle/HOL provides the additional advantage of being a fully fledged Higher Order Logic with a rich set of theories available. This enables modeling of the chain of trust through Delegation Signatures (DS) which is an important part of the DNSsec protocol. After the preliminaries in Section II, we present in Section III, the model of the Delegation Signature (DS) authentication for DNSsec followed by the results proved in Isabelle/HOL in Section IV. We conclude with related work and a discussion. The Isabelle/HOL sources are available in full online at https://sites.google.com/site/floriankammueller/home/resources.

II. Preliminaries

A. DNS and DNSsec

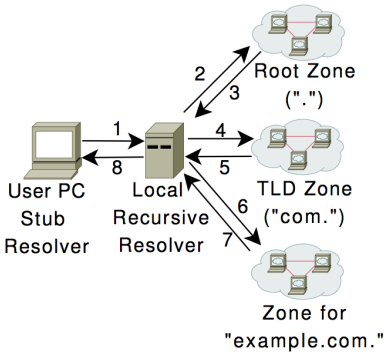
The DNS protocol organises how alphanumerical internet addresses, for example `www.mdx.ac.uk` are translated into numerical IP addresses, for example `162.192.10.1` [10]. This translation must clearly be performed in a correct way, otherwise some malicious server in the internet could provide a translation for an alphanumerical to an IP address of his own choice thus leading the client to connect to this address [3].

The way the DNS has been designed – as the entire internet protocol – is such that "spoofing" is very easily possible: data packets contain a source IP address which can be changed since packets are not encrypted[1]. Now, the DNS protocol is a very simple two-way protocol based on IP where a local recursive resolver (LRR) contacts a DNS server and asks for an address translation. In response the server sends back an IP package to the recursive resolver containing the translation of the query into an IP address. This two-way protocol is repeated about three times as illustrated in Figure 1 on a sample run. The repeated "recursive" process is due to the way internet IPv4 addressing is organised into *zones* of three or four groups of address chunks, e.g. `www.mdx.ac.uk`. For each of those zones, a DNS server is the authority and an LRR contacts first the root DNS authority server which in turn delegates his initial query by giving him a partial IP address translation that allows the LRR to follow up the query with the DNS server of the next zone [4]. Figure 1 depicts a typical flow of DNS name queries through the zones.

The only content in a DNS query and its corresponding response that ties those two packets together is a 16-bit transaction number commonly called TXID [4]. As for the source address it is trivial for the attacker to forge this TXID if he sees the DNS request sent by the client [4].

The attacks resulting from this lack of authentication and integrity assurance in the DNS protocol are the well-known

[1]Protocol extensions, like IPSec, do in fact encrypt message contents and may even encrypt source and destination addresses, but this requires the IPSec specific infrastructure in place, which is not normally given in IP traffic.



Reply	RRSets in DNS Reply	RRSets added by DNSSEC
3	*"com. NS a.gtld.net."* *"a.gtld.net. A 192.5.6.30"*	"com. DS" "RRSIG(DS) by ."
5	*"example.com. NS a.iana.net."* *"a.iana.net. A 192.0.34.43"*	"com. DNSKEY" "RRSIG(DNSKEY) by com." "example.com. DS" "RRSIG(DS) by com."
7	"www.example.com. A 1.2.3.4"	"example.com. DNSKEY" "RRSIG(DNSKEY) by example.com." "RRSIG(A) by example.com."
8	"www.example.com. A 1.2.3.4"	

Fig. 1. DNSsec repeatedly applies simple two-way query protocol [4].

cache-poisoning and the resulting man-in-the-middle attack [1]. These attacks have been known for some time but more recently the highly publicised variation on DNS cache poisoning discovered by Dan Kaminsky has attracted a lot of attention probably because several actual exploits of this attack on DNS servers run by ISPs redirected customers of these ISPs from popular web sites to attack sites [7].

DNSsec [2] is proposed as a solution to this issue. This protocol adds origin authentication based on public key cryptography as the main method to the simple two way protocol of DNS. The idea is simple: in asymmetric cryptography (or public key cryptography as it is more commonly known) the proprietor A of a private key K_A^{-1} can use this key to encipher a message. This cipher-code can then be used as a signature proving the integrity of the enciphered message: since anyone has access to the corresponding public key K_A, it is possible to test the signature by deciphering it with the public key and comparing the outcome with the original message:

$$M \stackrel{?}{=} \{\{M\}_{K_A^{-1}}\}_{K_A}$$

where we denote the application of a (specified yet here abstract) cryptographic algorithm to message M with key K as $\{M\}_K$.

DNSsec applies these ideas very straightforwardly: a LRR sends a DNS query out to a DNS server. The server sends back a signed response which contains the query and the correct resolution to an IP address signed by the DNS server's private key. The LRR uses the DNS server's public key to decode the response and check whether it is authentic. If this is the case, it can rely on the returned IP address resolution.

B. Delegation Signatures (DS)

However, how does the LRR know that the DNS server's public key really belongs to the server? This authentication of the public key is a general issue in public key infrastructures for which certificates are used. Certificates are attestations that a specific public key K_A belongs to a specific principal A. Technically such an attestation is a signed pair $\{(K_A, A)\}_{K_C^{-1}}$ certifying that the elements of the pair (K_A, A) belong together. The signature by $K_{C^{-1}}$, the private key of the certification authority, proves that C attests the relation between K_A and A.

This basic concept is used for the Delegation Signatures (DS) that are also added in DNSsec. They can be seen in Figure 1 on the right. For example, "`RRSIG(DS) by .`" represents a hashed signature of the server's public key by the root key. The DS build a so-called chain of trust, i.e., parent zones sign the public keys for their child zones. Assuming an initial trust in a global DNSKEY of the root zones, this extends the trust in the public keys of all zones.

The security of this DS authentication process is the main result of our paper.

To pave the way for a smoother understanding of our formalisation of DNSsec, we give the description of the protocol in "Alice and Bob" notation first. Let the zones of the IP be denoted as $z_0, \ldots, z_n$ where z_0 is the root zone, and let S_j be serving zone z_j with public and private keys K_{S_j} and $K_{S_j}^{-1}$, respectively. Then, the DNSsec protocol may be represented for each $j < n$ as follows.

$$\begin{array}{lcl} \text{LRR} \to S_j & : & \text{A} \\ S_j \to \text{LRR} & : & (K_{S_{j+1}}, \{\text{A}, \text{ip(A)}, \text{H}(K_{S_{j+1}})\}_{K_{S_j}^{-1}}) \end{array}$$

The identifier A represents the alphanumeric IP address, ip(A) maps this to a sequence of numbers of the next zone server S_{j+1} with public key $K_{S_{j+1}}$. The latter is hashed with a cryptographic hash H for efficiency. After receiving this message, LRR can use K_{S_j} to first decrypt the ticket in the second part and obtain the hash $\text{H}(K_{S_{j+1}})$. The hash cannot be inverted, but, since the hash algorithm H is a publicly known algorithm, e.g. SHA-256, LRR can apply it to $K_{S_{j+1}}$ that has been received as first part of the message in clear to check that this equals $\text{H}(K_{S_{j+1}})$. If this test succeeds, LRR believes that the public key $K_{S_{j+1}}$ belongs to S_{j+1} and ip(A) is A's address since the packet is signed with the private key $K_{S_j}^{-1}$ of the previous server S_j and LRR trusts that S_j has this key. This trust has been established in the same process in step $j-1$. Initially, the trust to the root's public key K_{S_0} must be provided by so-called "out-of-bounds" measures. DNSsec repeats these two steps three time; the final step, i.e. $j = 3$, just omits the first part of the second message.

Besides origin and DS authentication, the DNSsec protocol also provides in the more recent version defined in RFCs 4470 "DNSSEC Hashed Authenticated Denial of Existence" informally called NSEC3. The reason is that the DNSsec protocol allows adversaries to enumerate all the names in a zone by following the so-called NSEC chain. Although this is not an attack on DNS itself it could allow an attacker to map network hosts by enumerating the contents of a zone. NSEC3 simply avoids this by introducing a default "NSEC3" reply that is given in case a host is not in a zone thereby avoiding information flows. The expressiveness of Isabelle/HOL allows modelling this default message. This extension is possible in our model but beyond the scope of this paper.

C. The Inductive Approach to Security Protocol Verification

The interactive theorem prover Isabelle/HOL [11] implements classical higher order logic (HOL) for the modelling of application logics. Inductive definitions and datatype definitions can be written in a way close to programming languages. Semantic properties over datatypes can be formalised in a simple equation style by primitive recursion and are strongly supported by automated proof procedures based on rewriting, automated simplification, as well as externally coupled dedicated provers.

The inductive approach to security protocol verification by Paulson [12], the designer of the Isabelle system, picked up on the hype generated by the earlier model checking approach to security by Lowe [9]. In comparison, the inductive approach is more laborious as it requires human interaction, but it is unrivaled in its expressiveness which allows proofs beyond the ones that are usually done in model checkers.

Based on the inductive approach in the interactive theorem prover Isabelle/HOL [11], we show how the protocol DNSsec can be modelled and important properties are proved. The advantage of the inductive approach over model checking lies in its expressiveness. Although proofs in Isabelle/HOL are not performed automatically but have to be provided by the user, this increased expressiveness allows modelling protocols less abstractly than in model checking. A high level of abstraction may easily lead to an oversimplification and consequently overlooking attacks.

1) Attacker Model, Events, and Traces: The principals are expressed by a datatype definition guaranteeing their distinctiveness. We assume a server, a number of friendly principals, and a spy. That is, in our model the attacker is explicitly modeled.

```
datatype agent = Server | Friend nat | Spy
```

The attacker can forge messages using all components he can derive from previous traffic. The inductive operators characterize the constituents of a protocol's messages (set `parts`), messages the attacker can extract from a protocol trace (set `analz`), and messages that the attacker can build (set `synth`).

Protocols are defined by inductive definitions describing the behaviour of principals taking part in the protocol. Behaviours are sets of possible event traces. A trace is a list of communication events, such as interleaved protocol runs.

Compared to the inductive definitions for `synth` and `analz`, protocol definitions are thus of a different type: rather than specifying a message set, they specify the behaviour of the communicating principals as traces of *events* defined as a datatype comprising different types of protocol communication events. The main type of event is that an agent sends a message to another agent: the constructor `Says` takes three arguments of types `agent`, `agent`, and `msg` and returns one result of type `event`. The other constructors of the datatype `event` are `Gets` and `Notes` to specify the reception and storing of messages.

Defining a protocol in the inductive definition consists of defining a set of traces of events representing all possible runs of the specified protocol. The attacker's behaviour is added by including Fake messages into traces. The analysis first derives the knowledge he can extract from the protocol (`analz`) and the messages he can synthesise (`synth`). This characterizes the attacker's behaviour and allows verification of security properties.

Following the Dolev-Yao model [5], the `Spy` gets to know everything that is communicated along any channel. To this end, the inductive approach models a function `spies` that effectively reconstructs event traces into sets of messages. The central part of this definition (omitting a few technical details and the `Notes` and `Gets` cases) is as follows.

```
spies (ev # evs) =
  case ev of
       Says A' B X ⇒ insert X (spies evs) ...)
       ...
```

I.e., for any trace `ev # evs` its `spies`-set contains a message `X` if it was sent in event `ev`.

III. DNSsec Verification

In this section we first summarize the results on origin authentication and fixing replay attacks by nonces that we obtained for the protocol DNSsec by applying the inductive approach in Isabelle/HOL [8]. We then present in detail the formalisation of the Delegation Signature protocol for DNSsec showing the proved DS authentication property.

A. Origin Authentication and Replay Attack on DNSsec

We can use the inductive approach to prove *origin authentication of DNSsec*. Using a model of the protocol reflecting the communication steps between LRR and the DNSsec servers as introduced in Section II-A, the expressiveness of Isabelle/HOL allows us to explicitly represent public key infrastructure and corresponding security properties. Thereby, we can prove that in each step the returned IP addresses correspond to the requested IP addresses. However, this proof relies on the authenticity of the public key, i.e. that it really belongs to the server and is not inserted by an intruder. This authenticity of the relationship between the queried server and its public key is achieved by the chain of trust that can be built from Delegation Signatures. We are going to prove its authenticity

in this paper and thereby finally complete the proof of origin authentication.

However, there is another issue. The DNSsec specification as given in RFC3833 is not specific about the use of timestamps or nonces. There is a flaw (at least) in this specification but possibly also in general in (other implementations) of DNSsec: the server response can be replayed. We can formally prove the replay attack possibility on DNSsec. If the Server at some point in a run of the DNSsec protocol utters a (correct and signed) response, the Spy – who can record this – can *at any time afterwards* resend this Server response: replay attack.

This replay attack may seem pointless because it contains a correct IP address and a signature. But – as is usual with security attacks – the significance of the attack depends on the context. Since there is no boundary on the time between the first response and the replay, the attacker may have hijacked the IP address of the query in between the original response and the time of attack. The replay can then take effect if a RR requests the same alphanumeric address A to direct it to the (now) compromised site ip(A) – even with a correct Server signature.

To fix this bug in the protocol, the general cure against replay attacks would help: timestamps. However, we apply and prove correct an equally effective but more general version of DNSsec with *nonces*. Nonces may in principle be guessed so they must be sufficiently large. However, compared to timestamps they do not require synchronized clocks thereby avoiding a serious issue in distributed systems. By a simple extension of the DNSsec protocol with nonces we make the protocol safe against replays.

B. DNSsec Delegation Signatures in Isabelle/HOL

Given that the request resolver can trust the signature, origin authentication is thus provided by the DNSsec protocol. However to enforce this trust in the public keys of the servers, we need to verify the Delegation Signatures (DS) as introduced in Section II-B. We formalise the DNSsec protocol with delegation signatures in Definition DS_auth with the inductive definition package as illustrated in Figure 2. As can be seen

```
inductive_set DS_auth :: event list set
where
  Nil: [] ∈ DS_auth
| Fake: ⟦ evsf ∈ DS_auth;
          X ∈ synth (analz (spies evsf)) ⟧ ⟹
         Says Spy RR X # evsf ∈ DS_auth
| DS1: ⟦ evs1 ∈ DS_auth; RR ≠ Server ⟧ ⟹
       Says RR Server (Number A) # evs1 ∈ DS_auth
| DS2: ⟦ evs2 ∈ DS_auth;
         Says RR' Server (Number A) ∈ set evs2 ⟧
       ⟹
         Says Server RR' (Crypt(priK Server)
             {Number A, ip(A), Hash(Key A)})
         # evs2 ∈ DS_auth
```

Fig. 2. Inductive definition for DNSsec

there, this inductive definition specifies a set of event traces, i.e. lists of events. The initial trace is empty; it represents the beginning of all possible runs of the protocol (rule Nil in Figure 2). The Spy can analyse and synthesize from what he "spies", i.e. the set of things he knows (see previous Section). The Spy can then say all these things since he is an agent as well. This is implemented in the rule Fake. The local request resolver RR addresses the trusted Server to resolve an address for him. This address is called A and is the alphanumeric form of the address, e.g. www.mdx.ac.uk, represented as Number(A). The recursive resolver RR is assumed to be different from the Server to eliminate self requests from traces. The Server responds with the Key and the IP address ip(A) of the queried server address A. Keys are modelled in our approach as part of the Message datatype. The constructor Key is consequently an injective function which gives us that keys are different for different addresses.

```
Key :: address ⇒ nat
```

The Server encrypts the address Number(A), its IP address ip(A) together with the Hash of the public key of the child zone with his private key to ensure that this reply can be verified by RR' to originate from Server. A few noteworthy global facts about the specification of DS_auth are as follows.

- The principals Spy and Server are fixed constants of the inductive approach.
- By contrast, the principals RR and RR' are variables. They can be instantiated by arbitrary agents, for example, Spy or Server (the reason for excluding the latter possibility explicitly is to avoid meaningless traces of self-communicating servers).

IV. Proved Properties

To build up some infrastructure for proving the security of DS_auth with the inductive definition, we prove some characteristic lemmas, before we show authenticity of DS in lemma server_response_publik_key_is_authentic.

A. Basic Lemmata

A key lemma proves that for any trace evs that respects the protocol DS_auth, the private key of the Server is not known to the Spy.

```
Spy_sees_not_server_priK_DS: evs ∈ DS_auth ⟹
                Key (priK Server) ∉ parts (spies evs)
```

To prove such a lemma, the infrastructure of the inductive approach provides strong support. In fact, the lemma can be proved almost fully automatically after the induction scheme has been set up manually by the following command sequence.

```
by (erule DS_sec.induct,auto)
```

This command sequence is a concise proof script advising Isabelle/HOL to apply the induction rule that is generated (and proved automatically) for the inductive definition DS_auth (as defined in Figure 2). The resulting two proof obligations for the induction base and step can automatically be resolved using the proof procedure auto invoking various proof techniques and simplification procedures.

B. Delegation Signature Authenticity

A key fact about authenticity of the server response is now characterized by the lemma `server_msg_is_authentic_DS`. It characterizes authenticity: any response of the `Server` to `RR` has been preceded by the proper request of `RR`, i.e., the same address `Number A` as in the request is hashed and signed with the `Server`'s private key in the response.

```
lemma server_msg_is_authentic_DS:
⟦ (Says Server RR (Crypt (priK Server)
   {Number A, ip(A), Hash(Key(A))}) # evs)
   ∈ DS_auth
⟧ ⟹  Says RR Server (Number A) ∈ set evs
```

The proof of this lemma is simply performed by induction inversion over the definition `DS_auth`. This is one of the strengths of inductive definitions: rule inversion makes use of a basic implicit property of inductively defined sets: any of their elements can *only* be created by those rules enabling an exhaustive case analysis and inverse reasoning for elements of inductive sets.

Strengthening the previous lemma corresponds to an integrity result: the `Server` signature shows that the hash of the key that is returned is authentic. I.e., if some arbitrary but fixed number `pk` is returned within a signed response corresponding to the protocol step `DS2` then this `pk` *must* correspond to the correct public key of the requested address `A`.

```
lemma server_response_publik_key_is_authentic:
⟦ (Says Server RR (Crypt (priK Server)
  {Number A, ip(A), Hash pk }) # evs) ∈ DS_auth
⟧ ⟹  pk = Key(A)
```

To clarify the significance of this result a bit more we consider its simple logical contraposition: if some agent `S` signs and sends a "second" message, and this contains an "incorrect" IP address, this `S` has not been the `Server`.

```
lemma False_key_not_Server:
⟦ (Says S RR (Crypt (priK S)
  {Number A, ip(A), Hash pk }) # evs) ∈ DS_auth;
  pk ≠ Key(A)
⟧ ⟹ S ≠ Server
```

These lemmata show that origin authentication is guaranteed by the protocol as specified in the inductive definition `DS_auth`.

V. Conclusions

A. Related Work

The DNS protocol has been formalised with the model checker Murϕ [4]. This work describes existing attacks but is not very explicit on the formalisation of the protocol. It comes up with a list of practical recommendation on how to implement or deploy the DNSsec protocol. Those recommendation have reinforced the work of others and have since been addressed in their experimental (attack-based) scrutiny of DNSsec, e.g. [6]. Our model should have a similar impact, in particular as it adds an entirely new level of expressivity due to the use of Higher Order Logic. Our recent paper [8] provides more detail on the underlying inductive formalisation. Compared to this earlier formalisation, the current paper enriches the model to a fully fledged DNSsec-model with Delegation Signature (DS) signature chain extension. In the previous work we only addressed a simplified protocol not proving the chain of trust given by the DS signatures.

B. Discussion

An advantage of modelchecking compared to interactive theorem proving is that it is a "push-button" technology: the verification is fully automatic. Consequently, model checking needs to abstract to be decidable losing expressivity in the models. The state space needs to be finite and logics are only propositional with modalities like time or beliefs. Model checking performs a proofs by complete state exploration which immediately becomes computationally infeasible with larger state spaces. The number of states grows exponentially with the number of variables representing a state. This state-explosion problem prevents more expressive modelling and analysis.

Our approach of using Higher Order Logic with Isabelle/HOL and the inductive approach to security protocol verification is first of all much more labour intensive. Proofs are all done with human interaction, i.e. by hand. However, most of the proofs for DNSsec could be performed almost fully automatically. This is because Isabelle/HOL already has a very powerful set of automated proof procedures such as `auto`. In addition, Paulson's inductive approach offers a suitable infrastructure of general purpose lemmas for security protocols, e.g. agents, messages, and the sets `synth`, `analz`, etc. Therefore automation of proofs is largely possible and we can concentrate on the main steps of the proofs of theorems almost as in paper proofs. Moreover, since Higher Order Logic is very expressive, any kind of reasoning is possible, for example, theorems about keys, prime numbers, or time stamps can be explicitly performed in Isabelle/HOL as seen here.

References

[1] "Bind security advisory. dns cache poisoning issue ('kaminsky bug'), https://www.isc.org/sw/bind/forgery-resilience.php," 07/08/2008.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Rfc 4033: Dns security introduction and requirements," 2005.

[3] D. Atkins and R. Austein, "Threat analysis of the domain name system," in *(DNS). RFC 3833, Internet Engineering Task Force*, 2004.

[4] J. Bau and J. C. Mitchell, "A security evaluation of dnssec with nsec3," in *NDSS*. The Internet Society, 2010.

[5] D. Dolev and A. C. Yao, "On the security of public key protocols," in *SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. IEEE, 1981.

[6] A. Herzberg and H. Shulman, "Security of patched dns," in *ESORICS*, Vol. 7459, LNCS Springer, 2012.

[7] D. Kaminsky, "It's the end of the cache as we know it," August 2008.

[8] F. Kammüller, Y. Kirsal-Ever, and X. Cheng. DNSsec in Isabelle – Replay Attack and Origin Authentication. *Systems, Man, and Cybernetics, SMC'13*. IEEE 2013.

[9] G. Lowe, "Casper: A compiler for the analysis of security protocols," in *Computer Security Foundations Workshop (CSFW '97)*. IEEE 1997.

[10] P. Mockapetris, "Domain names - concepts and facilities," in *STD 13, RFC 1034*, 1987.

[11] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*. Vol. 2283, LNCS Springer, 2002.

[12] L. C. Paulson, "The inductive approach to verifying cryptographic protocols," *Journal of Computer Security*. Vol. 6, no. 1-2, 1998.