Enabling Secure On-line DNS Dynamic Update

Xunhua Wang, Yih Huang
Department of Computer Science
George Mason University
Fairfax, VA 22030 USA
{xwang4,huangyih}@cs.gmu.edu

Yvo Desmedt
Department of Computer Science
Florida State University
Tallahassee, FL 32306 USA
desmedt@cs.fsu.edu

David Rine
Department of Computer Science
George Mason University
Fairfax, VA 22030 USA
drine@cs.gmu.edu

Abstract

Domain Name System (DNS) is the system for the mapping between easily memorizable host names and their IP addresses. Due to its criticality, security extensions to DNS have been proposed in an Internet Engineering Task Force (IETF) working group to provide authentication. In this paper, we point out two difficulties in the current DNSSEC (DNS Security Extension) standards in the handling of DNS dynamic updates: 1) the on-line storage of a zone security key, creating a single point of attack for both inside and outside attackers, and 2) the violation of the role separation principle, which in the context of DNSSEC separates the roles of zone security managers from DNS server administrators. To address these issues, we propose a secure DNS architecture that is based on threshold cryptography. We show that the architecture adheres to the role separation principle without presenting any single point of attack. Our experimental results reveal that, in terms of signature computation times, our architecture incurs negligible performance penalty when using RSA/MD5 signatures but significant overhead when using DSA signatures. It is our belief that the high level of security that can be achieved by the proposed architecture far outweighs its potential overhead, especially in critical DNS zones, such as the `.com` *zone.*

1 Introduction

The Domain Name System (DNS) is a distributed database used in the Internet to map easily memorizable host names to their respective IP addresses [17]. The DNS name space is organized hierarchically. The top-level domain includes `.com`, `.edu`, `.org`, two-letter country codes from ISO-3166, and so forth [18]. Second level domain names typically designate individual institutions (for example, Yahoo Inc. is assigned a second level domain name `yahoo` under the top-level domain `com`). A *name server* is a program that holds the domain name information regarding a subtree (except the part that is further delegated) in the DNS hierarchy, called the *zone* of the server. Several predefined properties, or *resources*, can be associated with a given domain name. Exemplary domain name resources include IP address and alias. The association of a domain name with a resource is called a *resource record* (RR). The most important RR of a domain name is the "type A" RR, which contains the host IP address of the domain name. All the RRs pertaining to the domain names within a zone are stored in a *master file*, which is maintained by the *primary* name server of that zone. Each zone also supports zero or more *secondary* name servers, which obtain RRs from the primary server. Secondary servers are intended to reduce the workload of the primary server; they send appropriate RRs to clients in response to queries but are not involved in the maintenance of the master file.

Unfortunately, the DNS, a critical infrastructure component of the Internet, was designed without security considerations. In particular, the original DNS architecture provides no way for a client to authenticate a received RR. This loophole enables many security attacks [1, 22, 24]. For example, the man-in-the-middle attack allows a malicious third party to intercept the query message originated from a client and return an incorrect RR to the client. By providing an incorrect IP address for the requested domain name (for instance, `www.amazon.com`), a malicious third party can cause dis-

ruption of services and/or loss of business to the entity that owns the domain name (Amazon.com, Inc. in the example).

In response to the above concerns, the *DNS Security Extension* (DNSSEC) is proposed by an Internet Engineering Task Force working group [7]. The DNSSEC provides RR authentication by the use of digital signatures. In DNSSEC, each zone is equipped with a public/private key pair. The private key is used to digitally sign all the RRs in that zone. The response to a DNS query will also include the requested RRs and a *SIG RR* (signature RR), which contains the digital signature of the requested RRs. The zone public key, used to verify signatures, is disseminated by means of *KEY RRs*.

A major security principle of the DNSSEC is the role separation [21]: it differentiates the roles of the DNS zone manager from the DNS server administrator. In DNSSEC, it is the DNS zone manager, not the DNS server administrator, who is responsible for the security of a zone. A DNS server is just a place to publish the digitally signed DNS zone data. Thus, compromise of a secondary server or, if the zone private key is kept off line, even the primary server for a zone, will not necessarily affect the degree of assurance that a DNS client receives [7]. When zone data needs updating, the zone manager will digitally sign the new zone data off-line.

The idea of keeping a zone's private key off-line and computing the signatures of RRs off-line is based on the assumption that RRs in a DNS zone are relatively static. Indeed, in normal situations we can expect that the bindings between domain names and IP addresses do not change frequently. However, it has been pointed out that DNS dynamic updates, which enable real-time changes (that is, add, delete, or update) in name/address bindings, are useful under many circumstances [16, 25]. (For example, such an update allows a corporation to switch to a new domain name without waiting for the slow, manual processing of the name change.) For a dynamic RR update to take effect immediately and securely, the signature of the updated RR has to be computed on-line. It is worthwhile to note that although a DNS dynamic update requestor is communicating with a name server, the name server itself has no rights to update the zone data. Instead, it is the zone private key, instead of the name server's private key, that is needed in a dynamic update.

In the DNS dynamic update scheme by IETF, two modes are defined to achieve the above goal [6]. In mode A, the zone private key is kept off-line and any dynamic requests will be authorized by a *dynamic update key* that is kept on-line. However, in this mode, since the zone private key is not kept on-line, "zone transfer security is not automatically provided for dynamically added RRs, where they could be omitted, and authorization is not provided for the server denial of the existence of a dynamically added type" [6]. In this sense, this mode is not a genuine DNS dynamic update. In mode B, on the other hand, the zone private key is kept on-line at the zone primary name server, thus, any legitimate DNS dynamic updates can be in effect immediately.

Both of the above modes require that a zone security related private key be kept on line at the primary name server. This practice raises security concerns [6]. First, it makes the primary name server a single point of attack to an outside attacker. Should the primary server be compromised, the on-line private key will be exposed, rendering dynamic RRs insecure in mode A or, even worse, all RRs in the entire zone insecure in mode B. Second, as an insider, the name server administrator has access of this private key, compromising the role separation principle and making it possible for the name server administrator to abuse his/her power. (The importance of fending off insider attacks has been emphasized by the security community [10].)

The contribution of this paper is an alternate approach to the secure DNS dynamic update problem. In our proposed approach, through the introduction of zone-security servers and the application of threshold cryptography, the role separation principle holds and at the same time, genuine and secure dynamic updates are supported. The proposed architecture is also highly configurable, which allows it to achieve different levels of security. By giving concrete examples we show how our system can achieve intrusion tolerance against both outsider and insider attacks.

This paper is organized as follows: Section 2 reviews the related work and Section 3 discusses an important building block of our solution, namely, threshold cryptography. In section 4, we present our proposed architecture, discuss the security levels that can be achieved through configurations, and give some implementation details. Section 5 gives our experimental results and Section 6 discusses the integration and other issues of our proposed architecture. Conclusions of this study are given in Section 7.

2 Related Work

Bellovin pointed out the DNS authentication problem and explored how this flaw can be used to break into DNS-based systems and applications [1]. Schuba gave a detailed security analysis of the DNS system in [22]. Using the popular DNS software, BIND, as an example, Vixie discussed how to improve the security of various implementation aspects of DNS, such as caching [24]. DNSSEC can be found in [7] and dynamic update schemes are proposed in [6].

3 Threshold Cryptography

Threshold cryptography is a branch of cryptography that enables a group of n members, $1, 2, \ldots, n$, to act as a single

communication party, using one pair of public and private keys [2, 3, 4]. Similar to the traditional public key cryptography, the public key is known to the public. Unlike the traditional public key cryptography, the private key of the group, $K_{private}$, does not exist as a whole. Rather, each member i, $1 \leq i \leq n$, will be assigned ω_i key *shares* of the private key. (For each public key cryptography algorithm, a corresponding key sharing mechanism must be devised.) Thus the total number of key shares in a threshold cryptosystem is $\sum_{i=1}^{n} \omega_i$. To perform a cryptographic computation f (such as decryption, signing, identification, etc.) on message m with key $K_{private}$, b, $b \leq n$, group members will be required. Let $\{\pi_1, \pi_2, \ldots, \pi_b\}$, $1 \leq \pi_i \leq n$, $1 \leq i \leq b$, be such a group. Each member π_i will compute a partial result $g_{\pi_i}(m, k_{\pi_i})$ where k_{π_i} is one key share owned by π_i. Subsequently, the partial results are combined to produce the final result, i.e., $f(m, K_{private}) = C(g_{\pi_1}(m, k_{\pi_1}), g_{\pi_2}(m, k_{\pi_2}), \ldots, g_{\pi_b}(m, k_{\pi_b}))$, where C is the combination function. (This phase does not need to be performed by trusted entities.) Note that during this whole process the shared private key, $K_{private}$, is never reconstructed. Further, threshold cryptography requires that the shared private key cannot be reconstructed from any $t < b$ group members. Thus, a threshold cryptography scheme is characterized by its underlying public-key cryptography technology, a key sharing mechanism, the number of participants, n, and the threshold t. It is worth mentioning that, given a threshold cryptography scheme, b, the number of participants in a threshold cryptographic computation, is determined by n and t. The exact relationship among b, n and t varies from scheme to scheme. However, $b \geq t + 1$ must hold.

The researches on Threshold Cryptography mainly concentrate on the design of threshold RSA [5, 9, 11, 20] and threshold DSA [12, 13, 15]. Threshold RSA can be used for both decryption and digital signing while threshold DSA is only used for digital signing.

- Threshold RSA. Let (N, e) and d be a pair of RSA public and private keys, respectively, such that, e and d are relatively prime to $\phi(N) = (p-1)(q-1)$ and $e \times d = 1 \bmod \phi(N)$, where p and q are two primes and $N = p \times q$. In RSA, for both decrypting and signing, $f(m, d) = m^d \bmod N$.

  Desmedt, Di Crescenzo and Burmester described an elegant threshold RSA scheme in [5], where $b = t+1$. A simple example of this threshold RSA scheme is the case of $n = 3$ and $t = 1$ (thus $b = 2$). Let A, B and C be the 3 members. A will be assigned a random number, d_1, B will be given 2 values, d_2 and d_3, and C will be given 2 values, d_2 and d_4, where $1 < d_i < \phi(N)$ for $1 \leq i \leq 4$, $d = d_1 + d_2 \bmod \phi(N)$ and $d = d_3 + d_4 \bmod \phi(N)$. Thus, A has one key share and B and C have two key shares. Any two of them will be able to perform a cryptographic computation $f(m, d)$. For example, $f(m, d)$ can be computed by members A and B in two steps. First, A computes $\tau_A = g_A(m, d_1) = m^{d_1} \bmod N$, and B computes $\tau_B = g_B(m, d_2) = m^{d_2} \bmod N$. Second, the partial results produced in the first step are combined to produce $\tau_A \times \tau_B \bmod N = m^{d_1} \times m^{d_2} \bmod N = m^d \bmod N = f(m, d)$. Since C also possesses d_2, the above computation can also be performed collectively by A and C. Moreover, the above result can also be produced by $m^{d_3} \times m^{d_4} \bmod N$, which can be performed by B and C.

- Threshold DSA. Threshold DSA is more complex than threshold RSA due to the structure of the DSA algorithm [8]. Details are omitted here due to the space limitations. To our knowledge, the best result of threshold DSA is to have $t < n/2$ and $b = 2t + 1$ [12]. Moreover, the co-signing process requires more involved interactions among the b parties, compared to those of the threshold RSA. However, one advantage of threshold DSA over threshold RSA is that each of the participants receives exactly one key share (that is, $\omega_i = 1$, $1 \leq i \leq n$), which simplifies the task of key management.

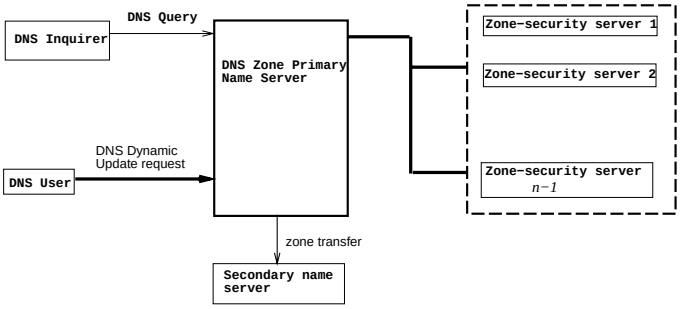

Figure 1. The proposed secure DNS architecture

4 The Proposed Architecture

In this section we present an architecture that supports genuine and secure dynamic updates and, at the same time, separates the zone manager role and the name server administrator role. Moreover, the architecture is highly configurable to achieve intrusion tolerance against outsider attacks and insider attacks.

4.1 The Architecture

Depicted in Figure 1 is the proposed secure DNS architecture. In our architecture, we assume that there exist $(n-1)$, $n \geq 2$, machines in a given DNS zone that are under the control of the zone manager, but not under the control of the name server administrators. We call these machines the *zone-security servers*. Using a threshold cryptography scheme with $n > t \geq 1$, the zone private key is shared among the $(n-1)$ zone-security servers and the primary name server. Let $b > t$ be the number of zone private key shares needed in the computation of the signature of an RR. Since $b \geq 2$, any change to the zone data will need the cooperation of at least one zone-security server; the name server administrator will have no way to modify the digitally signed zone data. Thus, the role separation principle holds. Moreover, the above architecture enhances intrusion tolerance of DNS [26].

A DNS system is said to be *l-intrusion-tolerant* against an entity, E, if controlling l servers (including the zone-security servers and the DNS name servers, if applicable) that are outside E's control will not help E gain any knowledge of the zone private key. A DNS system is said to be intrusion tolerant against an outsider (insider) if it is at least 1-intrusion-tolerant against the outsider (insider). Our proposed architecture can be configured to be intrusion tolerant against both outsiders and insiders.

- Intrusion tolerance against outsider attacks. In our architecture, the zone private key cannot be recovered from any single location, thus, making the system intrusion tolerant against an outside attacker. That is, even when an outside attacker manages to corrupt l, $l \leq t$, of relevant servers (including the name servers and the zone-security servers), secrecy of the zone private key is still maintained.

- Intrusion tolerance against insider attacks. The presence of zone security servers and the necessity of their involvement in signature computations constitute a defense line against insider attacks, in particular, attacks from name server administrators. Clearly, a hostile name server administrator must break into at least t zone security servers (to get access to the respective key shares) in order to perform unauthorized RR signature computations.

RFC 2535 defines two types of SIG records, the DSA SIG and the RSA/MD5 SIG [7]. The DSA SIG uses the NIST DSA signature algorithm and the RSA/MD5 SIG uses the MD5 hash and the RSA algorithms. The important configuration and implementation aspects of our proposed architecture with respect to these two types of SIGs are given next.

Table 1. Example configurations in terms of t-n.

Security Level	RSA/MD5 SIG		DSA SIG	
	1-2	2-4	1-3	2-5
Intrusion tolerant against outsider attacks (Y/N)	Y	Y	Y	Y
Intrusion tolerant against insider attacks (Y/N)	N	Y	N	Y

4.2 Configurations

We now discuss the configuration of the proposed architecture, i.e., the selection of t and n values:

- The value of t (the threshold): Obviously, t must be greater than or equal to 1. However, if $t = 1$, there will be a single point of attack for the name server administrator. To achieve a high level of intrusion tolerance against the name server administrator, t should be no less than 2.

- The value of n (the number of group members): This value is determined by the underlying threshold cryptography algorithm. With threshold RSA, $n \geq t + 1$. With threshold DSA, however, the best known result is $n \geq 2t + 1$.

In Table 1, we give the representative configurations to achieve the above security levels in different application cases. Additional details regarding these configurations are provided below. In the discussion, the primary name server will be referred to as server 0 and the $(n-1)$ zone-security servers will be referred to as servers 1, 2, ..., $(n-1)$.

4.3 Some Details

The DSA SIG. In this case, for a t-n configuration (such as the 1-3 and 2-5 configurations), the zone manager will generate n key shares of the zone private key and distribute them to the n servers using a (t, n) Shamir secret sharing scheme [23]. When the zone data needs updating, $(2t + 1)$ servers are required to jointly compute the DSA SIG RR [12].

The RSA/MD5 SIG. Assume that, in RSA, the zone's private key is d and its public key is (e, N). We use the threshold digital signature scheme given in [5] and now discuss how the zone private key is shared and how to generate a RSA/MD5 SIG RR.

- The 1-2 case. In this case, the zone manager generates a random, d_1, $1 < d_1 < \phi(N)$, and compute

Table 2. Keys for 2-4 RSA/MD5 SIG

Server	Keys Assigned	Participating Servers			
		(0,1,2)	(0,1,3)	(0,2,3)	(1,2,3)
0	d_1,d_6	d_1	d_1	d_6	
1	d_2,d_6	d_2	d_2		d_6
2	d_3,d_5	d_5		d_3	d_3
3	d_4,d_5		d_5	d_4	d_4
Note		$d = d_1 + d_2 + d_5 \bmod \phi(N)$		$d = d_6 + d_3 + d_4 \bmod \phi(N)$	

d_2, $d_2 = d - d_1 \bmod \phi(N)$. Values d_1 and d_2 are sent securely to the primary name server and the zone-security server respectively. To generate a RSA/MD5 SIG of m, where m is the MD5 digest of an RR, these two servers will compute $m^{d_1} \bmod N$ and $m^{d_2} \bmod N$ respectively. Then any one of them can combine the partial results as $m^{d_1} * m^{d_2} \bmod N = m^d \bmod N$, the digital signature of m by the zone private key.

- The 2-4 case. To generate key shares, the zone manager generates 4 random numbers, d_1, d_2, d_3, d_4, where $1 < d_i < \phi(N)$ for $1 \leq i \leq 4$. He/she will also compute two numbers, $d_5 = d - d_1 - d_2 \bmod \phi(N)$ and $d_6 = d - d_3 - d_4 \bmod \phi(N)$. Subsequently, d_1 and d_6 are *securely* sent to the primary name server, d_2 and d_6 are *securely* sent to server 1, d_3 and d_5 are *securely* sent to server 2, d_4 and d_5 are *securely* sent to server 3, as shown in Table 2.

  To generate a RSA/MD5 SIG, any 3 of the 4 servers will be needed. For example, if the primary name server and the zone-security servers 1 and 2 are present (the (0,1,2) case in Table 2), the three servers will compute $m^{d_1} \bmod N$, $m^{d_2} \bmod N$, and $m^{d_3} \bmod N$ respectively. After that any one of them can combine the partial results to produce $(m^{d_1}) * (m^{d_2}) * (m^{d_3}) \bmod N = m^d \bmod N$, the digital signature of m by the zone private key. Other possibilities of computing the signature of m are summarized in Table 2.

5 Experimental Results

We implemented the proposed architecture on a platform that comprises several NT workstations interconnected by a 10Mbps local area network. Our implementation is based on the multiple precision integer package in [19]. Table 3 summarizes the experimental results. For comparison, Table 3 also gives the times to compute a SIG on one machine under the same condition, i.e., the same multiple precision integer package and the same platform. We also observe that the times given here are meaningful only for comparing the performances of our proposed architecture and the time to compute a SIG on one machine, since the performances of different multiple precision integer packages vary. For example, the multiple precision integer package GNU MP [14] has much better performance than [19].

Table 3. Experimental Results

	Zone Key Size (bits)	t-n	Time Cost, in seconds, to compute a SIG	
			Proposed Archtecture	On one machine
RSA/MD5	1024	1-2	2.172	2.0935
		2-4	2.110	
SIG	2048	1-2	15.9295	15.9408
		2-4	15.8311	
DSA	512	1-3	0.7906	0.1942
		2-5	1.371	
SIG	1024	1-3	1.4217	0.4646
		2-5	2.6999	

In our architecture, servers, including the name server and the zone-security servers, compute in parallel. Thus the only additional overhead of our proposed architecture in terms of time is the communications among the servers. Our results show that this overhead is largely negligible in threshold RSA. For DSA SIG, our approach is considerably costlier than an approach that is based on the conventional DSA algorithm, such as DNSSEC. However, it is our belief that the high security level of our approach outweighs its performance penalty, especially in critical DNS zones, such as `.com` zones.

6 Further Discussion

Lastly, we discuss several further enhancements to the proposed secure DNS architecture.

Integration of the DSA and RSA/MD5 configurations. In Section 4.2, we give different configurations to implement the DSA SIG and the RSA/MD5 SIG. Actually they can coexist in a single DNS zone. For example, to achieve intrusion tolerance against outsider attacks, we can adopt the 1-3 configuration for both DSA SIG and RSA/MD5 SIG. The only difference between them is that all 3 servers are required to update a DSA SIG while any 2 is enough to update a RSA/MD5 SIG. Similarly, for intrusion tolerance against the name administrator we can adopt the 2-5 architecture for both DSA SIG and RSA/MD5 SIG.

Surviving server failures. For the DSA SIG, the aforementioned configurations cannot survive server failures because all servers are required to participate. Survivability can be achieved by increasing n, for example, we can have

1-4 and 2-6 configurations for intrusion tolerance against outsider attacks and insider attacks respectively.

Excluding the name servers from signature computation. In order to have a uniform architecture for intrusion tolerance against both outsider attacks and insider attacks, the primary name server is used as a member in the computation of signatures. Indeed, for the intrusion tolerance against the name server administrator case, a simpler alternative from the management perspective is to assign *no* key shares to the primary name server. As such, all key shares will be assigned to the zone-security servers. In such case, for DSA SIG, 1-3, instead of 2-5, can be used to achieve intrusion tolerance against the name server administrator and, for RSA/MD5 SIG, 1-2, instead of 2-4, can be used to achieve intrusion tolerance against the name server administrator.

7 Conclusion

We have proposed an architecture in which genuine, secure DNS dynamic update is supported and the role separation of the zone manager and the name server administrator holds. The proposed architecture can be configured to achieve intrusion tolerance against both outsider attacks and insider attacks. Implementation details and experimental results about the DSA SIG and RSA/MD5 SIG are given. Our experimental results reveal that, in terms of signature computation times, our architecture incurs negligible performance penalty when using RSA/MD5 signatures but significant overhead when using DSA signatures. It is our belief that the high level of security that can be achieved by the proposed architecture far outweighs its potential overhead, especially in critical DNS zones, such as the `.com` zone.

8 Acknowledgement

The authors wish to thank the anonymous reviewers who helped to improve Table 3 and pointed out the related work in [22].

References

[1] S. M. Bellovin. Using domain name system for system break-ins. In *Proceedings of the Fifth Usenix UNIX Security Symposium*, Salt Lake City, UT, June 1995.

[2] Y. Desmedt. Society and group oriented cryptography : a new concept. In C. Pomerance, editor, *Advances in Cryptology, Proc. of Crypto '87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, California, U.S.A., August 16–20 1988. Springer-Verlag.

[3] Y. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4):449–457, July-August 1994. (Invited paper).

[4] Y. Desmedt. Some recent research aspects of threshold cryptography. In E. Okamoto, G. Davida, and M. Mambo, editors, *Information Security*, volume 1396 of *Lecture Notes in Computer Science*, pages 158–173, Tatsunokuchi, Ishikawa, Japan, September 1997. Springer–Verlag.

[5] Y. Desmedt, G. D. Crescenzo, and M. Burmester. Multiplicative nonabelian sharing schemes and their application to threshold cryptography. In J. Pieprzyk and R. SafaviNaini, editors, *Advances in Cryptology — Asiacrypt '94*, volume 917 of *Lecture Notes in Computer Science*, pages 21–32, Wollongong, Australia, November/December 1994. Springer-Verlag.

[6] D. Eastlake. *RFC2137 – Secure Domain Name System Dynamic Update*, April 1997.

[7] D. Eastlake. *RFC2535 - Domain Name System Security Extensions*, March 1999.

[8] N. I. for Standards and Technology. Digital signature standard (DSS), February 2000.

[9] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Tech. Report TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin–Milwaukee, April 1992. ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedtrsathreshold_92.ps.

[10] B. Fraser. *RFC2196 – Site Security Handbook*, September 1997.

[11] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In N. Koblitz, editor, *Advances in Cryptology — Crypto '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 157–172, Santa Barbara, California, U.S.A., August 18–22 1996. Springer-Verlag.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Advances in Cryptology — Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371, Zaragoza, Spain, May 12–16 1996. Springer-Verlag.

[13] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, ACM Symposium on Parallel Algorithms and Architectures, pages 101–111, Puerto Vallarta, Mexico, June 28 – July 2 1998.

[14] T. Granlund. *GNU MP*. Source code available from `http://www.gnu.org/manual/gmp/index.html`.

[15] S. K. Langford. Threshold DSS signatures without a trusted party. In D. Coppersmith, editor, *Advances in Cryptology — Crypto '95*, volume 963 of *Lecture Notes in Computer Science*, pages 397–409, Santa Barbara,California, U.S.A., August 27–31 1995. Springer-Verlag.

[16] C. Liu. Securing an Internet name server. Presentation, 1999. Available at http://www.acmebw.com/papers/securing.pdf.

[17] P. Mockapetries. *RFC1035 – Domain Names - Implementation and Specification*, November 1987.

[18] J. Postel. *RFC1591 - Domain Name System Structure and Delegation*, March 1994.

[19] Pr0duct Cypher. *PGPTools*. Source code available from `ftp://nic.funet.fi/pub/crypt/utilities`.

[20] T. Rabin. A simplified approach to threshold and proactive RSA. In H. Krawczyk, editor, *Advances in Cryptology, Proc. of Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 89–104, Santa Barbara, California, USA, August 23-27 1998. Springer-Verlag.

[21] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information System Security*, 2(1):105–135, February 1999.

[22] C. Schuba. Addressing weaknesses in the domain name system protocol. Master's thesis, Purdue University, August 1993.

[23] A. Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.

[24] P. Vixie. DNS and BIND security issues. In *Proceedings of the 5th Usenix Security Symposium*, Salt Lake City, UT, June 1995.

[25] P. Vixie, S. Thomson, Y. Rekhter, and J. Vound. *RFC2136 - Dynamic Update in the Domain Name Systems(DNS Update)*, April 1997.

[26] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *Proceedings of the 8th USENIX Security Symposium*, pages 79–91, 1999.