Verifying Global Invariants in Multi-Provider Distributed Systems

Sridhar Machiraju
UC Berkeley

Randy H. Katz
UC Berkeley

ABSTRACT

Confidentiality is an important requirement that restricts information sharing between multiple providers in inter-domain routing and, more generally, in any *Multi-Provider Distributed System (MPDS)*. However, sharing such confidential information can make these systems more robust by enabling the verification of global system invariants. For instance, undesirable interaction between intra-domain and inter-domain routing can be prevented by verifying system invariants involving confidential intra-domain information of neighboring domains. In the literature, it is generally assumed that global system invariants of MPDSs involving confidential information cannot be verified. In this paper, we demonstrate that this is *not true* by constructing proof-of-concept protocols that verify two such invariants relevant for robust inter-domain routing. Our work exposes a hitherto unexplored portion of MPDS design space that has the potential for making such systems more robust.

1. INTRODUCTION

An important property of inter-domain routing is that competitive concerns force individual providers to hide most of the intra-domain information that might be necessary for robust routing. Such confidentiality concerns apply to a variety of information including routing policies, link properties, link failures, etc. Though measurements can reverse-engineer capacities [3], policies [27] etc. to a reasonable extent, we argue that measurement inaccuracies and overhead make these techniques useful more for end-to-end decisions than for infrastructure-based decisions.

Experience with inter-domain routing has shown that the lack of crucial information can affect the robustness of the system. For instance, lack of knowledge about congestion in neighboring Autonomous Systems (ASs) may cause an AS to make bad intra-domain traffic engineering decisions [8, 28, 31]. Similarly, in the absence of sufficient information on the policies of other ASs, an AS could use routing policy that leads to divergent routing [19]. Such problems can be solved by having the ASs verify suitably defined global invariants [31]. However, since these invariants use confidential intra-domain information, most existing solutions to such problems remove the need for verifying global invariants by limiting local autonomy [8, 14, 29]. We argue that confidentiality-preserving verification of invariants is useful in any *Multi-Provider Distributed System (MPDS)* since providers in these systems typically have confidentiality requirements. Examples include systems for caching [4], computing [12] and storage [20].

Winick et al. [31] used problem-specific aggregation techniques to limit the amount of shared information in verifying global invariants. However, their solution does not preserve the confidentiality of such shared information. In this paper, we propose the use of provably secure solutions for *Secure Multi-Party Computations (SMPC)* [1] to verify global invariants. Since solutions to solve arbitrary SMPC problems are highly inefficient [32], we take the approach of constructing efficient problem-specific solutions. Such an approach was shown to be feasible for information sharing between databases [2]. We believe that this approach is also feasible for MPDSs because many invariants in MPDSs are likely to involve simple arithmetic operations. Simple arithmetic operations can be realized securely and efficiently using homomorphic cryptosystems [25]. We also believe that the commercial out-of-band relationships between providers in an MPDS (e.g., peering relationships in inter-domain routing) simplify the threat model that needs to be considered. Specifically, these relationships can be leveraged to address issues such as arbitrary protocol termination and key exchange.

We demonstrate our approach by designing proof-of-concept protocols that verify global invariants ensuring safe traffic engineering and verifying policy safety in inter-domain routing. We believe that the generality of our techniques make them applicable in any MPDS when the invariants involve simple arithmetic and out-of-band relationships exist. We emphasize that our goal is not to advocate the use of the global invariants that we use. Rather, our aim is to expose a powerful class of cryptographic techniques that can be employed to

address robustness issues in MPDSs.

The organization of this paper is as follows. In Section 2, we define global invariants, clarify our assumptions and introduce homomorphic cryptosystems. In Section 3, we develop a protocol to verify a global invariant that ensures safe intra-domain traffic engineering. In Section 4, we use a global invariant for conflict-free routing policies to motivate the hardness of verification with more than two providers. We mention related work in Section 5. We discuss future directions and conclude in Section 6.

2. OVERVIEW

In this section, we first formally define global invariants in MPDSs and our goal. Then, we discuss the assumptions we make, including the threat model. Finally, we introduce homomorphic cryptosystems.

2.1 Global Invariants

In an MPDS, a global invariant refers to a triplet $G = (P, I, f())$. Here, P is a set consisting of more than 1 provider. I is a set of inputs, each of which belongs to one of the providers in P. Also, each provider in P has at least one input in I. $f()$ is a boolean-valued function that is evaluated on I. For the MPDS to function properly, $f(I)$ must be *true*. We refer to testing this condition as verifying the invariant G. In this paper, our goal is to illustrate how G can be verified when some of the inputs (in I) are confidential and will not be revealed by the owner to other providers.

2.2 Assumptions and Threat Model

We assume that, given a secure protocol, providers would provide confidential information as input. We believe that this is true because the maintenance of global invariants is necessary for the functioning of the system (and hence, to the vital interests of the participants). However, providers may lie about their inputs to influence the behavior of other providers, e.g., by causing the invariant to be violated. Addressing this is outside the scope of this paper and preventing this may be specific to the problem being solved. Mechanisms belonging to the area of *Distributed Algorithmic Mechanism Design (DAMD)* [9] can be used to address these issues. DAMD mechanisms are complementary to our protocols; While they ensure that providers do not lie about their inputs, secure protocols can be used to implement these mechanisms without revealing the inputs.

Any protocol to verify a global invariant $f(I)$ cannot guarantee complete confidentiality of I. This is because the output, $f(I)$, leaks some information about I to the participants. Hence, a secure protocol should prevent any more information leakage. However, a malicious provider could abruptly terminate a protocol thereby causing asymmetric information leakage. We believe that such arbitrary termination is not a problem because providers in MPDSs are likely to possess an out-of-band relationship, e.g., commercial peering relationships in inter-domain routing. Since such relationships can be used to impose penalties on the malicious provider, we assume that abrupt termination is not a threat. We assume that the specific cryptosystems used in our protocols are strong enough that the ciphertexts leak negligible information about the plaintexts. Hence, we do not consider the threat of eavesdroppers and third parties.

2.3 Homomorphic Encryption

Next, we describe homomorphic cryptosystems that allow us to perform simple arithmetic (addition and multiplication) on encrypted values. A cryptosystem for which $E()$ represents the encryption operations, is said to be (additive) homomorphic if $E(m_1)E(m_2) = E(m_1 + m_2)$. We only use additive homomorphic cryptosystems such as Paillier's [25] in this paper. The time complexity of Paillier's is similar [25] to El Gamal [7]. to We note that multiplicative homomorphic cryptosystems such as El Gamal[7] may also be useful, in MPDSs. The following are important properties of (additive) homomorphic cryptosystems:

- $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$, the homomorphic property.
- $(E(m))^k = E(mk)\ \forall\ k$. A special case is $k = -1$.
- The ciphertexts before and after performing one or more of the above operations can be used to deduce the operation. For instance, $E(m) = E(-m)^{-1}$. In such cases, $E(-m)$ can be multiplied by a *blinding factor* so that the operation cannot be deduced.

3. SAFE INTER-DOMAIN TRAFFIC ENGINEERING

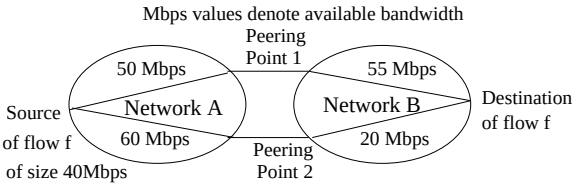

Figure 1: Shifting flow f from peering point 1 to 2 might seem advantageous without knowledge of available bandwidth in B.

It is known [8, 29, 31] that traffic engineering within an AS A can adversely affect a neighboring AS B by overloading links in B. For instance, in Figure 1, A's operator might change the exit point of flow f from 1 to 2 based on available bandwidth in A only. As we can

Variable	Description	Private to
$\{d_i\} 1 \leq i \leq D$	Unique destination prefixes	None
$\{p_j\} 1 \leq j \leq P$	Peering points between A and B	None
$\{e_l\} 1 \leq l \leq N$	links in B	B
$u_{i,j,l}$; $\vec{U}_{i,j} = (\{u_{i,j,l}\}_l)$	1 if route to dest. d_i from p_j uses link e_l, 0 otherwise	B
b_l; $\vec{B} = (b_1, \ldots b_N)$	available bandwidth on link e_l of B	B
$\delta_{i,j}$; $\vec{\Delta} = (\{\delta_{i,j}\}_{i,j})$	Proposed change in traffic to d_i entering B at p_j	A

Table 1: Notation used for Safe Traffic Engineering.

see, this has the effect of overloading the 20Mbps link in B. Preventing this from happening requires that, s_l, the available bandwidth on any link e_l in B after traffic engineering by A, should be non-negative. Using the notation in Table 3,

$$s_l = b_l - \sum_{(i,j)} \delta_{i,j} u_{i,j,l} \geq 0 \quad \forall\ l \tag{1}$$

Formally, the global invariant that needs to be satisfied is $G = (\{A, B\}, \{\vec{\Delta}, \vec{B}, \{\vec{U}_{i,j}\}_{i,j}\}, f())$ where

$$f(I) = [\vec{B} - \sum_{(i,j)} \delta_{i,j} \vec{U}_{i,j} \geq \vec{0}] \tag{2}$$

A similar invariant was proposed in [31]. They verified the invariant by reducing the information revealed using problem-specific aggregation. In contrast, we propose the use of a provably secure class of cryptographic techniques. Nevertheless, the aggregation techniques used in [31] can be used along with our protocol to achieve better scalability. The approach proposed in [8, 29] is to remove the need to verify the invariant by bringing each $\delta_{i,j}$ value close to 0. This is an alternative approach that has the disadvantage of restricting the set of traffic engineering choices that A can make.

3.1 Verifying the Invariant

We now describe a protocol that verifies the invariant specified above. We assume that A uses homomorphic public-key encryption and provides its public key to B using its out-of-band relationship. E_A and D_A denote the corresponding encryption and decryption operations. The basic idea behind the following protocol is that B can compute the s_l values by the homomorphic property. A can decrypt these values to verify the invariant. The protocol is as follows. (1) A sends $E_A(-\delta_{i,j})\ \forall\ (i,j)$ to B. (2) B calculates $E_A(s_l)\ \forall l$. It can do this by computing $E_A(b_l)$ and multiplying it with all $E_A(-\delta_{i,j})$ for which $u_{i,j,l}$ is 1. The product is $E_A(s_l)$ by the homomorphic property. (3) The various $E_A(s_l)$ values are sent to A. (4) Using the decryption operation D_A, A can verify if all N values of s_l are positive.

Clearly, E_A ensures that B learns nothing about A's inputs. However, the above protocol leaks information on the s_l values to A.

3.2 Reducing Information Leakage

Next, we modify the above protocol to prevent A from knowing anything but $f(I)$. For ease of exposition, we assume the existence of a cryptographic primitive $\mathcal{C}_A$ that allows A to determine $sign(x)$(1 if $x \geq 0$, -1 otherwise), without determining the value of x. We describe $\mathcal{C}_A$ later. In the above protocol, B can invoke $\mathcal{C}_A$ on each s_l value to prevent A from knowing the s_l values. However, this provides A with information on $sign(s_l)$. To prevent such leakage too, we let B invoke $\mathcal{C}_A$ on $-s_l$ with a probability of half. The results of these invocations are finally converted into *cong*, the number of negative s_l values, using the homomorphic property. Finally, $\mathcal{C}_A$ is invoked on *cong*. The complete modified protocol is as follows:

- A sends $E_A(-\delta_{i,j})\ \forall\ (i,j)$ to B.
- B calculates $E_A(s_l)\ \forall l$.
- If B invokes $\mathcal{C}_A$ on $E_A(s_l)$, A can determine $sign(s_l)$. To prevent this, B invokes $\mathcal{C}_A$ on $E_A(s_l r_l)$ where r_l is ± 1 with equal probability. $E_A(s_l r_l)$ can be calculated from $E_A(s_l)$ using homomorphism.
- To prevent B from knowing $sign(s_l r_l)$, A sends $E_A(sign(s_l r_l))$. Since A can pre-compute $E_A(\pm 1)$, there is no computational complexity.
- Using the homomorphic property, B calculates $E_A(cong)$ by computing $E_A(\sum_l sign(s_l r_l) r_l - N)$. Observe that, since $r_l = \pm 1$, $sign(s_l r_l) r_l = sign(s_l)$. Clearly, $cong < 0$ iff at least one link gets congested.
- B invokes $\mathcal{C}_A$ on $E_A(cong)$ so that A can determine $sign(cong)$ and hence, if the proposed traffic matrix change is safe or not.

Since no plaintext is ever revealed to B, no information is leaked to it. The plaintexts revealed to A are $sign(s_l r_l)$ which leaks no information since r_l is random. A does obtain information on N, the number of links in B in the third step. B can prevent this by sending encryptions of a few dummy values in addition to $E_A(s_l r_l)\ \forall\ l$. Thus, the above protocol does not reveal anything other than $f(I)$. Note that A (or B) can misbehave so that the final boolean value is a function of I different from $f()$. Addressing this is out of the scope of this paper.

3.2.1 Cryptographic Primitive $\mathcal{C}_A$

The cryptographic primitive $\mathcal{C}_A$ that allows A to determine $sign(x)$ without knowing x is closely related to

Yao's millionaires' problem [32]. In our threat model, we do not consider arbitrary termination of the protocol. This makes it possible for us to design our primitive to be much simpler than previous solutions [5]. We also assume that the absolute value of x is less than a small number, x_{max}. This is justified because the s_l values typically take, say, 10000 values from 1Mbps till 10Gbps.

Our primitive uses commutative encryption [2]. Two encryption functions f_1, f_2 are commutative if $f_1(f_2(x)) = f_2(f_1(x))$. For suitably chosen prime p, any two discrete-log based encryptions [7] defined as $f_y(x) = x^y (mod) n$ satisfy this property because $(x^{y_1})^{y_2} mod(p)$ is $(x^{y_2})^{y_1} (mod) p$. The basic idea behind $\mathcal{C}_A$ is for A and B to choose commutative encryption functions f_A and f_B using the out-of-band mechanism. These are used by A to verify the membership of x in the set of integers from $[0, x_{max}]$ securely.

- Given $E_A(x)$, B calculates $E_A(x+r)$ for some random r using the homomorphic property and sends it to A.
- B calculates the set $Pos_B = (f_B(r), f_B(r+1), \ldots, f_B(r + x_{max}))$ and sends a randomly permuted Pos_B to A.
- A decrypts $x + r$ and sends $f_A(x + r)$ to B.
- B calculates $f_A \circ f_B(x + r)$ and sends it to A.
- A checks if it belongs to the set $Pos = (f_A \circ f_B(r), \ldots, f_A \circ f_B(r + x_{max}))$.

B cannot determine x since E_A is secure and it is computationally hard for B to calculate $x+r$ from $f_A(x+r)$. A knows $x+r$ but the random r prevents it from knowing anything about x. Also, the hardness of inverting f_B prevents A from knowing the value of r from Pos_B. To prevent either party from using previous comparison operations, A and B should frequently change the commutative encryption functions used.

Though the above protocol requires the encryption of $x_{max} + 1$ values (with $f_A \circ f_B$), this can be pre-computed. Also, efficient compression tools such as compressed Bloom filters [24] can be used to efficiently store such pre-computed vectors. Hence, $\mathcal{C}_A$ essentially requires one encryption, decryption and three exponentiations. Thus, the running time of the modified protocol in Section 3.2 is dominated by the $O(DP)$ encryptions in the first step, the $O(M + N)$ decryptions in the third step and N invocations of $\mathcal{C}_A$. The number of large-integer operations in calculating s_l is cN where c is the number of links each flow would traverse. Assuming values of D, P, N, c of about 500, 10, 1000 and 10 [26] respectively, we see that a few thousand encryption/decryption operations need to be performed. These can be done in a few seconds with specialized hardware.

4. POLICY-INFLICTED DIVERGENCE

As shown in [19], BGP routing policies can interact to cause divergent routing. In general, checking for divergence is NP-complete. The solution proposed in [14] is for domains to locally constrain their policies. It is generally thought [18] that this is preferable because of the hardness of checking for divergence and confidentiality of inter-domain policies. In this section, we show that the confidentiality of policy is not a problem. Due to the lack of prior work exploring easily verified policy specifications, we motivate a simple specification and develop a protocol to verify it. We emphasize that our contribution is to demonstrate that confidentiality concerns are no hurdle to verifying policy safety; Our goal is not to advocate that ASs use our policy specification.

4.1 A Simple Invariant

The solution proposed in [14] is to have providers constrain their policies such that (1) The customer-provider directed graph is acyclic. (2) A route through a customer should always be preferred over a route through a peer or provider. (3) A route through a provider or peer should not be exported to another provider or peer. We

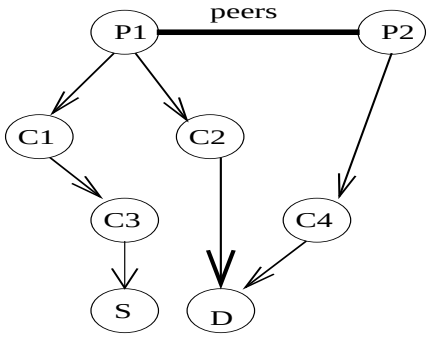

Figure 2: For a destination D, $P1$ may prefer to export to $C1$ a path through $P2$ over a path through $C2$.

believe that, in the future, such local constraints alone may not satisfy the requirements of ASs. We illustrate this using the provider-customer directed graph shown in Figure 2. The link between Provider $P1$ and its customer $C2$ is congested. To alleviate this congestion, $P1$ would like traffic to destination D from S to transit $P2$, and not $C2$. The scenario described above requires $P1$ to violate the local constraints above and prefer a longer peer route over a customer route.

Observe that, in the long term, $P1$ in Figure 2 may find it economically advantageous to upgrade the link to $C2$. However, until that time, it might have to resort to choosing $P2$, if possible. This motivates us to consider the possibility of *small deviations* from local constraints. Specifically, for each destination D, policy safety might

be ensured even if a small number of ASs violate one or more of the local constraints. For instance, the system in Figure 2 is convergent if $P1$ alone deviates from the second policy constraints. To see this, consider the effect of $P1$ changing its policy from preferring $C2$ to $P2$. The effect of this change would be:

- A peer/provider of $P1$ to which $P1$ did export the original route through $C2$ would see a withdrawal from $P1$. Hence, the effect of the policy change is equivalent to failures of the links from $P1$ to these ASs. This is known to be safe.
- $P1$ may not have exported the original route to some peers/providers. They do not receive any new update. This is because a route through a peer is not advertised to other peers/providers.
- Direct/indirect customers of $P1$ might receive the newer route and may prefer this route (through their providers) and advertise it to their customers only. Since the provider-customer graph remains acyclic, this is safe. In terms of activation sequences [17], the stable state is achieved by activating ASs from $P1$ downward.

4.2 The New Global Invariant

The small deviation mentioned above would not cause route divergence if at most one AS per destination prefix uses the deviant policy of preferring a peer over a customer. This can be formalized as a global invariant $G = (P, I, f())$ for every destination prefix. Here, P is the set of all N Autonomous Systems. I consists of $r_p(\forall p \in P)$ where r_p is 1 if AS p follows a deviant policy to reach the corresponding destination prefix and 0 otherwise. The system is safe if $\sum_p r_p = N$. The invariant G can be verified without revealing I as described in the following protocol:

- A threshold variant of Paillier's [11] is used so that each AS possesses a share of the private key and only a large number (say, more than half) of the ASs can together decrypt ciphertexts. The public key is known to all ASs and hence, each of them can calculate $E(m)$. The shared secret key can be reused many times and hence, can be constructed out-of-band.
- Each AS p advertises $E(r_p)$.
- Each AS multiplies all the advertised values to calculate $E(\sum_p r_p) = E(n)$, where n is the number of deviant ASs.
- Decryption of $E(n)$ can be done in a distributed manner[11]. Iff n is less than 2, the deviant AS can follow the deviant policy. Otherwise, all deviant ASs must abandon the deviant policy.

The distributed nature of the secret key ensures that the r_p values are not revealed to others. Note that, after the protocol, the number of deviant ASs is known. Each instance of the above protocol requires one distributed encryption and decryption. This can easily be done in 30 seconds. By running the above protocol for 30 destination prefixes in parallel every 30 seconds, policy safety for almost all destination prefixes can be verified every day. The requirement of only a single deviant AS per destination prefix is probably too strong. We are currently investigating weaker (and easily verifiable) conditions. Also, the above protocol requires that all ASs share a key which may not be possible. We are currently investigating policy specifications and verification techniques that can best leverage the bilateral peering relationships.

5. RELATED WORK

Many homomorphic cryptosystems have been proposed in cryptographic literature. Some examples include the El Gamal cryptosystem [7] and Paillier's [25]. The general problem of *Secure Multi-Party Computations (SMPC)*[32] has been the subject of extensive research [1]. Recent work has applied solutions to specific computations to solving practical problems. Du et al. [6] study a variety of specific secure two-party computations. The solution proposed in [5] assumes a third party to solve Yao's problem since their threat model includes arbitrary termination. Cryptographic protocols for constructing secure set operations are presented in [2]. Commutative encryptions are also used in [2].

Many recent proposals propose MPDSs for a variety of tasks, apart from inter-domain routing, such as caching [4], computing [10, 12] and storage [20]. In the context of inter-domain routing, interactions between intra-domain and inter-domain routing have been explored in [8, 28, 29, 31]. The solutions proposed in [23, 31] verify global invariants using problem-specific aggregation techniques that reduce the amount of shared information. These do not address the fundamental question of working with confidential information. All other prior work [8, 28, 29] restrict intra-domain traffic engineering so that it does not adversely affect inter-domain routing. The possibility of policy-conflicts in inter-domain routing was shown in [15, 16]. Gao et al. [14] designed routing policy guidelines that ensured safety by restricting local autonomy. Gao et al. [13] extended the local constraints to allow backup routing using information such as the the number of backup links on a path. In essence, they introduce a distributed global invariant that reveals information on backup links. The computation complexity of verifying policy safety for an arbitrary policy specification was shown in [18, 19]. The hardness of verifying policy safety due to the confidentiality of policies was alluded to, in [14, 18].

6. CONCLUSIONS AND FUTURE WORK

In this paper, we argue that solutions to secure multiparty computations can be used to verify global invariants involving confidential information in MPDSs. We design two proof-of-concept protocols to demonstrate our approach. Our protocols verify two global invariants, one for safe traffic engineering and another to verify policy safety. We do not claim that these invariants are the most important invariants that need to be verified or that our techniques are the most optimal way of verifying them. Our protocols expose a powerful class of cryptographic techniques that can make MPDSs more robust without sacrificing significant local autonomy when (1) Out-of-band commercial relationships between providers can be leveraged to simplify the threat model and for key distribution. (2) Global invariants involve simple arithmetic operations. Whether other invariants can be verified easily is an open question. SMPC can potentially be used for global decisions other than verifying invariants. For instance, some of our ongoing work [22] is aimed at developing solutions that allow more general forms of cooperative routing than considered in this paper. Other global decisions in MPDSs arise in the context of content distribution networks [4] and intrusion detection [21, 30]. Future work also needs to address fundamental limitations in using SMPC for MPDSs.

7. ACKNOWLEDGMENTS

We thank Karthik Lakshminarayanan, Mukund Seshadri, Weidong Cui and Lakshminarayanan Subramanian for useful discussion and comments about this paper. We also thank the anonymous reviewers for their insightful comments.

8. REFERENCES

[1] Secure Multiparty Computations. `http://www.tcs.hut.fi/~helger/crypto/link/mpc/`.

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of ACM SIGMOD*, 2003.

[3] D. "Allen B. Using Pathchar to Estimate Internet Link Characteristics. In *Proc. of ACM SIGCOMM*, 1999.

[4] L. Amini, A. Shaikh, and H. Schulzrinne. Effective Peering for Multi-provider Content Delivery Services. In *Proc. of IEEE INFOCOM*, 2004.

[5] C. Cachin. Efficient Private Bidding and Auctions with an Oblivious Third Party. In *Proc. of ACM CCS*, 1999.

[6] W. Du. *A Study of Several Specific Secure Two-party Computation Problems*. PhD thesis, Purdue University, West Lafayette, Indiana, 2001.

[7] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *Proc. of Advances in Cryptology (CRYPTO)*, 1984.

[8] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for Interdomain Traffic Engineering. *ACM SIGCOMM CCR*, October 2003.

[9] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proc. of Dial-M*, 2002.

[10] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science*, 2001.

[11] P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. *Lecture Notes in Computer Science*, 1962, 2001.

[12] Y. Fu et al. SHARP: An Architecture for Secure Resource Peering. In *Proc. of SOSP*, 2003.

[13] L. Gao, T. Griffin, and J. Rexford. Inherently Safe Backup Routing with BGP. In *Proc. of IEEE INFOCOM*, 2001.

[14] L. Gao and J. Rexford. Stable Internet Routing without Global Coordination. In *Proc. of ACM SIGMETRICS*, June 2000.

[15] R. Govindan, C. Alaettinoglu, G. Eddy, D. Kessens, S. Kumar, and W. Lee. An Architecture for Stable, Analyzable Internet Routing. *IEEE Network Magazine*, January/February 1999.

[16] T. Griffin, F. B. Shepherd, and G. T. Wilfong. Policy Disputes in Path-Vector Protocols. In *Proc. of ICNP*, Toronto, Canada, November 1999.

[17] T. Griffin and G. T. Wilfong. A Safe Path Vector Protocol. In *Proc. of IEEE INFOCOM*, March 2000.

[18] T. G. Griffin, F. B. Sheperd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM TON*, 10(2), April 2002.

[19] T. G. Griffin and G. Wilfong. An Analysis of BGP Convergence Properties. In *Proc. of ACM SIGCOMM*, 1999.

[20] J. Kubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proc. of ASPLOS*, November 2000.

[21] P. Lincoln, P. Porras, and V. Shmatikov. Privacy-Preserving Sharing and Correlation of Security Alerts. In *Proc. of USENIX Security Symposium*, August 2004.

[22] S. Machiraju and R. H. Katz. Reconciling Cooperation with Confidentiality in Multi-Provider Distributed Systems. Technical Report UCB//CSD-04-1345, 2004.

[23] R. Mahajan, D. Wetherall, and T. Anderson. Interdomain routing with Negotiation. Technical Report CSE-04-06-02, 2004. University of Washington.

[24] M. Mitzenmacher. Compressed Bloom Filters. In *Proc. of ACM PODC*, 2001.

[25] P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Proc. of Eurocrypt*, 1999.

[26] K. Papagiannaki, N. Taft, and C. Diot. Impact of Flow Dynamics on Traffic Engineering Design Principles. In *Proc. of IEEE INFOCOM*, March 2004.

[27] N. Spring, R. Mahajan, and T. Anderson. Quantifying the Causes of Path Inflation. In *Proc. of ACM SIGCOMM*, August 2003.

[28] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proc. of ACM SIGMETRICS*, June 2004.

[29] R. Teixeira, A. Shaikh, T. Griffin, and G. M. Voelker. Network Sensitivity to Hot-Potato Disruptions. In *Proc. of ACM SIGCOMM*, September 2004.

[30] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. Large Scale Malicious Code: A Research Agenda, 2003. DARPA sponsored report, `http://www.cs.berkeley.edu/~nweaver/large_scale_malicious_code.pdf`.

[31] J. Winick, S. Jamin, and J. Rexford. Traffic Engineering between Neighboring Domains, July 2002. `http://www.research.att.com/~jrex/papers/interAS.pdf`.

[32] A. C. Yao. Protocols for Secure Computations. In *Proc. of FOCS*, 1982.