Techniques for Policy Enforcement on Encrypted Network Traffic

Y. Sinan Hanay and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{hanay,wolf}@ecs.umass.edu

Abstract—**Most large-scale data communication networks are built from multiple autonomous subnetworks, which are managed by different administrative entities. In many practical environments, information about traffic policies is considered proprietary and is not disclosed by network operators. However, some operational scenarios require routers within a network to check if traffic matches a particular policy that is provided by another entity. In our work, we present several algorithms of how to represent policy databases and how to perform policy checks without explicitly disclosing the total set of policies. This privacy-preserving set operation extends related work, which has assumed that parties trust each other. Our analysis shows that the proposed policy checks can be implemented efficiently in realistic systems.**

I. Introduction

Network operators use policies to express what data traffic is permitted to traverse a certain set of network nodes. Such policies can permit or restrict traffic from a particular source (e.g., by specifying an IP address prefix), traffic related to a particular application (e.g., by specifying a destination port number), etc. With an increasing number of malicious sources of traffic (e.g., botnets) and an increasing number of applications that impact network performance (e.g., peer-to-peer downloads), the number of policies that need to be enforced on a node can reach the order of thousands. Due to changes in operational dynamics of the network, the set of policies that is active at a given point in time may change during runtime.

The set of active network policies for a network contains much implicit information on its provider's operational practices. Thus, these policies are typically considered proprietary information. Most network providers do not want to make their policy database available to the public and their competitors. However, in practice there are scenarios, where a party needs to verify that traffic matches the policy of a competing party. As we show in Section III, such scenarios can arise in conventional and military networks. These scenarios are characterized by two criteria:

- Set Operation on Policy Database: The operation that needs to be performed is a set intersection. That is, it needs to be checked if a particular packet matches any of the policies in a policy database. More generically, it needs to be checked if an item (i.e., the packet) is in a set (i.e., the policy database).
- Mutual Mistrust: The parties involved do not trust each other. Thus, the party that provides the policy database does not want to provide this database in cleartext. The party that performs the set operation does not trust the party providing the policy database to perform the necessary computations. Thus, there is mutual mistrust.

Previous solutions to related problems have not considered cases where parties mistrust each other and do not want to reveal details about a successful policy match (e.g., what policy matched for particular traffic).

In this paper, we present several mechanisms for these policy operations that can be performed by parties that do not trust each other. We show that the use of a third party that serves as a trusted authority can overcome mutual mistrust. The specific contributions of our work are:

- The design of solutions that introduce a trusted third party,
- Four specific policy checking mechanisms that preserve privacy, and
- An analysis of the effectiveness and space and time costs of each mechanism.

The remainder of the paper is organized as follows.

This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

Section II discusses related work. Section III provides a more formal problem statement as well as realistic example scenarios where this problem appears in practice. Section IV provides an overview of our approach to solving the problem and a brief background on the techniques used. Section V presents the four policy checking mechanisms that we consider as well as an evaluation of their complexity. Section VI summarizes and concludes this paper.

II. Related Work

In computer networks, policies are typically expressed as deny/permit rules for connections. These connections are specified by a 5-tuple of network protocol fields including IP source and destinations, source and destination port, and transport layer protocol identified. As packets traverse the network, routers can check which rules they match by using packet classification algorithms. A survey of these classification algorithms can be found in [12].

As we discuss below, once policies can be expressed as 5-tuple permit/deny rules, a policy check can be implemented as a set operation. Privately determining the set intersection of multiple parties has been studied previously. Agrawal et al. provide a process that determines whether two parties share common elements in several rounds [1]. Pinkas et al. present an approach where a polynomial representation of the set is used [9]. Kissner and Song improve the efficiency of polynomial evaluation by utilizing mathematical properties of these polynomials [14].

A closely related problem is secure policy reconciliation. In the secure reconciliation problem, multiple parties try to agree on a set of policies that is consistent with all parties' policies. Meyer et al. extend the privacy preserving set intersection methods described in [9] to achieve secure policy reconciliation [17]. A performance evaluation of these privacy preserving protocols under different configurations is presented [18].

The methods providing set cardinality given in [9], [14], [17] use properties of homomorphic encryption schemes and parties succeed in preserving privacy of their set intersection. Instead of having encryption of each element of the set, a previously agreed value is chosen for encryption, and at the end of the protocol, the chooser counts the occurrences of encryptions of the chosen value. This approach is not feasible for our case since it requires several rounds of computation by the involved parties. The main difference in our work from previous solutions is that, in our setting parties have to decide intersection in a single round. Our work aims at carrying the necessary information in a single network packet.

III. Problem Statement

The problem we consider in our work can be stated as follows:

> Given a database of policies, determine if a packet matches any policy without revealing the database content or which specific policy matches the packet. This policy check should be performed in a single round without per-packet interactions between the policy provider and the policy checker.

More generally, this problem can be rephrased as a set operation problem: Given a set, determine if an element is a member of that set while maintaining the privacy of the set and the element. This problem is a special case of the set intersection problem, where one set consists of only one element. Before explaining where this problem occurs in practice, we introduce notation to simplify the discussion and a more formal statement of trust relationships.

A. Notation

We use the following notation to formalize the problem statement. This notation is used throughout the rest of the paper:

- Ω: Set of all possible policies; the number of elements in Ω is n (i.e., $|\Omega| = n$).
- P: A set of allowed policies $P \subseteq \Omega$ (i.e., the policy database); the number of elements in P is m (i.e., $|P| = m$). We denote $P = \{p_1 \dots p_n\}$.
- Q: The representation of P that protects the information contained in P. A policy p_i is denoted as q_i in Q.
- t_i: The traffic that matches policy p_i in P.
- u_i: The representation of t_i that preserves the privacy of t_i and that is used for policy checks against Q.

In an environment, where parties trust each other (i.e., the information in the policy database P and traffic t_i do not need to be protected), we can perform the following simple test to see if traffic is valid (i.e., may be forwarded):

$$valid(t_i) \iff (p_i \in P). \tag{1}$$

However, when aiming to protect the information contained in P, we perform the validity check using the "protected representation" of traffic and policy database:

$$valid(u_i) \iff (q_i \in Q). \quad (2)$$

Determining what an effective "protected representation" looks like is the topic of this paper.

For simplicity of discussion, we make the following three assumptions:

- We assume a deny-by-default environment, where policies indicate what traffic is permitted. Thus, we do not have to deal with a combination permit and deny policies. This assumption is realistic for some deny-by-default networking environments [2], [19]. If necessary, this work can be extended to consider a combination of permit and deny policies, each represented in its own database.
- We assume that there is a one-to-one relationship between t_i and p_i (i.e., $t_i \leftrightarrow p_i$). For practical networking policies based on 5-tuples with ranges, this relationship is more difficult to determine. In such a case, each field of the 5-tuple can be handled as a separate policy. For simplicity of discussion, we assume that a simple one-to-one relationship holds.
- We assume each policy p_i (and traffic t_i) can be represented by unique integer value.

B. *Trust Relationship and Confidentiality*

The premise of our work is that the involved parties do not trust each other. We consider two parties in our work: the entity that provides the policy database P ("policy provider") and the entity that performs the check if t_i matches a policy in P ("policy checker"). Specifically, the parties make the following assumptions related to trust (or mistrust in our case):

- The policy provider does not want the policy checker to be able to extract the set of policies p_i that are contained in the database P. Thus, the policy checker should only have access to the protected representation Q.
- The policy provider does not want the policy check to be able to determine which policy p_i matches any particular network traffic. Thus, the policy checker should only have access to a packet's protected representation u_i.
- The policy checker does not trust the policy provider to perform the computation of a policy check in its stead. Thus, the policy checker cannot simply hand traffic t_i to the policy provider and ask if it matches a policy in P.
- Both parties trust a third, independent party that acts as intermediary to perform the privacy protecting computations (e.g., translation from P to Q and thus can ensure protection of P towards the policy provider and a valid representation of Q towards the policy checker.

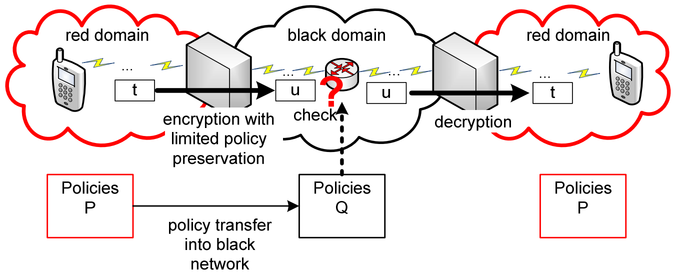

Fig. 1. Policy Check Scenario in Red/Black Network. Systems in the red domain are trusted and main carry sensitive data in cleartext. Systems in the black domain are not trusted and may carry data only in encrypted form. How to enforce policies on encrypted traffic in the black network is the problem addressed in this paper.

We illustrate the practical relevance of such a trust relationship through an example below.

Related to trust relationships is the question of what it means to protect the policy database P. This protection is considered as maintaining confidentiality of P with respect to policy checker when using the protected representation of traffic (i.e., u_i instead of t_i). Thus, an adversary (e.g., the policy checker) should not able to get meaningful information about P by eavesdropping on network traffic (i.e., observing u_i). We define meaningful information as:

- Information regarding which policies p_i a flow u_i matches.
- Information regarding whether two packets u_i and u_i' belong to the same flow.

The first notion is covered through the protected representation of P as Q. However, the second notion requires a more complex solution. If an adversary monitors traffic, they should not be able to infer how frequently packets from the same flow are sent by simply checking how frequently u_i is observed.

C. *Example Scenario*

The following example scenario illustrate where the problem that we solve appears in practice. Our work has been motivated by a problem that occurs in a military networking environment. Related examples from outside the networking domain are presented in [8].

A typical configuration of a red/black network is illustrated in Figure 1. That figure also shows the policy database in its representations P and Q, as well as traffic that is represented as t and u. On the borders between the red and the black network, the transformations in the representations are performed. The systems that perform this translation can be simple VPN gateways [10] or military-grade High Assurance Internet Protocol Encryptors (HAIPE) [6].

The trust relationship in this environment is as follows:

- The red domain does not trust the black domain: The red domain does not want to make the policy database P available to the black domain (which would reveal information on what policies that are applied in a classified network). Also, the red domain does not want the black domain to know what traffic is sent (i.e., what policy p_i matches traffic t_i) since this information would reveal information about network usage between the red domains.
- The black domain does not trust the red domain: The black domain does not want to carry arbitrary traffic sent between the red domains. The black domain may be willing to carry some types of traffic between the red domains, but not all.

A practical scenario of such a relationship would be a joint operation by military forces from two allied countries. The red domain represents forces from one country that need to communicate using a second country's communication infrastructure (i.e., the black domain). The second country may be willing to carry some type of traffic (e.g., situational awareness information), but not other types (e.g., non-tactical information).

The challenge of such a scenario is that a naïve solution, where all information from the red domain is hidden from the black domain (e.g., through encryption), is not suitable. The black domain requires enough information to be able to perform policy checks on traffic that is tunneled between red domains. This means, we need to perform what we call "limited policy preservation" (as shown in Figure 1). This step translates t_i to u_i in such a way that a check of u_i against Q is possible within the black domain. The design of methods for limited policy preservation (while maintaining privacy) is discussed in the following section.

IV. Design and Techniques

With an understanding of the problem we are trying to solve and the trust relationships between parties, we turn towards the general design of our policy check mechanisms. We first introduce the need for a third party that is trusted. Then, we specify the requirement for our solution before we provide a brief background on the techniques used in the solutions presented in Section V.

A. Trusted Third Party

Due to the mutual mistrust of parties, it is necessary to introduce a third party, which is trusted by the other parties. Such an approach is common in security protocols (e.g., key distribution). In our system, the third party performs the following function:

- Translation of Policy Database P to Q: In this step, the policy provider's policy database is translated using the privacy preserving operations described below. The reason that this step should be performed by the third party (rather than by the policy provider) is that the third party can check the cleartext policies p_i against policies specified by the policy checker. In our example, the black domain may agree to forward specific types of traffic from the red domain but not others. The third party can ensure that Q contains only those policies that the black domain agrees to.
- Translation of Traffic t_i to u_i: In this step, traffic is translated into the representation used by the policy checker. In our example case, that is the operation performed by the gateway illustrated in Figure 1. The reason the third party should be involved in this step is that the red domain may cheat and incorrectly label traffic to ensure it gets forwarded through the black domain even though it does not match an agreed-upon policy. In practice, it is not feasible to have a third party placed on each border network systems. However, practical systems, like HAIPE devices, require National Security Agency (NSA) certification. During the certification process it could be detected if a system incorrectly labels traffic. Thus, the black network simply has to check that its border systems are certified devices.

Using this approach to establishing a minimum level of trust, we can achieve the goals laid out in the problem statement.

B. Limited Policy Preservation

For a successful policy check, a network packet needs to carry to necessary information to permit the evaluation of Equation 2. Together with the need for protecting the information about which traffic t_i the packet belongs to, we can identify two specific requirements:

- Robustness against Statistical Attacks: Different packets that belong to the same type of traffic t_i should look different when represented as u_i. This requirement ensures that the policy checker (or another attacker) cannot make statistical inferences.

In a practical system, there may be a limit on how many different representations of u_i for a single t_i are possible. Thus, this requirement may only hold over a certain window of time.

- Compactness to Fit into Single Packet: Since our problem focuses on the networking domain, we need to consider practical limitations on how much data can be packed into a packet header. Thus, compactness is of the representation of u_i is an important design criteria.

To address the first requirement, we use probabilistic encryption, where the encrypted representation of p_i changes with each packet. To address the second requirement, we use RSA accumulators to reduce the space requirement of u_i. The backgrounds for each of these techniques are described in the following two subsections.

C. Background: Probabilistic Encryption

Most common encryption schemes are deterministic in their nature. When encrypting the same data repeatedly, the same encrypted output is generated. In the context of our problem, that means that two packets from traffic t_i would have the same representation u_i and be vulnerable to statistical inference attacks. Another problem is that the encryption may not be secure for all possible messages. For example, the encryption of 0 and 1 using unpadded RSA yield the cleartext of the message. Also, under certain conditions some partial information can be derived about the message [16].

In contrast, probabilistic encryption algorithms encrypt the same cleartext differently each times. This approach to encryption was first proposed by Micali and Goldwasser [11]. There are other probabilistic cryptosystems have been proposed. We describe two of these systems in more detail.

1) Micali-Goldwasser Cryptosystem: This probabilistic encryption scheme proposed by Micali and Goldwasser [11] is the first of its kind. Data are encrypted bitwise. The cryptosystem's security is based on hardness of determining whether a residue is quadratic or not. The cyrptosystem uses the following processes:

- Key Generation: Determine two primes p and q, and public key N is defined as $N = p \times q$ and private key is p and q
- Message Encryption: Encryption is done bitwise. Represent message m as a string of bits (m_1, ..., m_n). For each bit m_i, a random value $y \in Z_N^*$ is generated. Output $c_i = y^{m_i} x^2 \bmod N$.
- Message Decryption: Decryption is done bitwise $(c_1, ..., c_n)$. For each i, using the prime factorization (p, q), determines whether the value c_i is a quadratic residue; if so, $m_i = 0$, otherwise $m_i = 1$. Output the message $m = (m_1, ..., m_n)$.

The drawback of this scheme is that bitwise encryption and decryption is not linearly parameterizable.

2) ElGamal Cryptosystem: ElGamal is a public key algorithm [7], where security is based on hardness of the decisional Diffie-Helman problem [4]. The same cleartext is encrypted differently each time depending on the random number, a parameter of the ElGamal cryptosystem. The process is as follows:

- Key Generation: Generate a prime q and a generator $g \in Z_q$, construct a cyclic group G. Choose a random $x \in Z_q$ and compute $h = g^x$. The public key is (G, q, g, h), and the private key is (G, q, g, x).
- Message Encryption: For cleartext message $m \in G$, choose a random $y \in Z_q$ and output ciphertext pair using public key (G, p, g, h) as $(c_1, c_2) = (g^y, h^y \times m)$.
- Message Decryption: For ciphertext pair (c_1, c_2), output message $m = \frac{c_2}{c_1^x}$ using private key (G, p, g, x).

In this scheme, the number of different possible encryptions for each message is equal to possible values that random number y can take. Note that defining y as a parameter requires modification of El-Gamal slightly. User defines which value y can take, instead of taking any element from the Z_q as y.

D. Background: RSA Accumulators

RSA accumulators were first presented in [3] and are used to compact large lists to a representative and small accumulated value. Using this representative value, a decision can be made on whether some element belongs to the list. The process is as follows:

- Accumulator Construction: The trusted third party computes the value of cryptographic accumulator $Acc(Q)$ as $Acc(Q) = a^{q_1 q_2 \cdots q_n} \bmod N$, where a is a random number. The trusted third party also calculates witnesses for all allowed policies $w_i = a^{q_1 q_2 \cdots q_{i-1} q_{i+1} \cdots q_n} \bmod N$. The trusted third party provides N and $Acc(Q)$ to the policy checker.
- Per-Packet Operation: Each packet carries policy q_i with its witness w_i.
- Policy Check: To check if u_i is present in Q, the policy checker computes if $w_i^{q_i} = Acc(Q) \bmod N$.

RSA accumulator is secure under strong RSA assumptions as shown in [15].

V. Mechanisms for Policy Checks

We consider four different mechanisms for representing the policy database and performing policy checks and discuss the tradeoffs between the solutions.

A. Mechanism I: Polynomial Evaluation

This is the approach that was followed in [14]. However a similar approach is presented in [9], [17] using homomorphic encryption schemes. All parties generate polynomial representations of their list in which the roots are their private list elements (if an element appears multiple times in a private list, then this element is a repeated root of that polynomial). Then every node encrypts the polynomial coefficients of their sets, randomize it and then all parties add their encrypted, randomized coefficients to form a combined polynomial which is an encrypted polynomial representation of all parties' lists. If a node wants to see whether some particular element is in the intersection, it evaluates the combined polynomial to see if this value is a root of the polynomial. If so, they conclude that the element is in the intersection.

We can describe this mechanism more formally for our scenario as follows: Given the policy database $P = \{p_1, p_2, ..., p_m\}$ and a packet from traffic i, t_i. The protected representation of the policy database is the randomized polynomial representation of P:

$$Q(x) = (x - p_1)(x - p_2)...(x - p_m)r_Q(x), \qquad (3)$$

where $r_Q(x)$ is a random polynomial. Assume t_i matches policies $\rho_1, \rho_2, ...\rho_j$. The gateway between the red domain and the black domain generates the polynomial representation of t_i as:

$$u_i(x) = (x - \rho_1)(x - \rho_2)\dots(x - \rho_j)r_i(x), \qquad (4)$$

where $r_i(x)$ is a random polynomial. (Note that from this point on we allow traffic to match multiple policies.) To check if the packet is matching a valid policy in Q, a router extracts $u_i(x)$ from the packet, adds it to $Q(x)$, and check if the resulting polynomial has a root:

$$valid(u_i) \iff \exists x : Q(x) + u_i(x) = 0. \qquad (5)$$

One problem is that we have only two (instead of multiple) parties in the typical setting of this problem, and the randomizing polynomials are more likely to have common roots, which cause false positives. This problem can be overcome by using higher degree polynomials and a bigger ring at the cost of higher checking times.

B. Mechanism II: Encrypted Lists

To address the problems with polynomial representation, one can make policies as encrypted lists available to the policy checker. This approach is similar to different search techniques on encrypted databases [5], [13], [14]. Each policy in database P is encrypted, and the list of the encrypted policies makes up Q. The same encryption mechanism is used to translate t_i to u_i. Since the key used in the encryption is not known to the policy checker, the content of P and traffic t_i remain protected. The main drawback is the vulnerability to statistical attacks (e.g., by observing periodic small packets with the same u_i value, the attacker can infer that it may be a voice-over-IP connection). To avoid this problem, multiple keys can be used to obtain multiple encryptions (for each policy and for each traffic). When translating from t_i to u_i the encryption key is chosen randomly from the set of keys. This approach increases the size of Q, but requires no additional space to represent u_i.

C. Mechanism III: ElGamal with Prime Numbers

An alternative to choosing from a set of encryption keys is to use a randomized cryptosystem. As discussed in Section III, ElGamal is one such system, where policies are encrypted using one of k different encryptions. Multiple matching policies for traffic require a representation as a list, which may become inefficient, or as an irreducible polynomial as discussed for Mechanism I. Using the latter approach, for each possible ciphertext pair (c_i, c_j) a prime number can be assigned. The policy checker maintains the product of these $k \times |Q|$ prime numbers. The policy check is done by checking whether this product is relatively prime to the value u_i, the encrypted policy carried within the packet. The disadvantage of this approach is that a big prime product needs to be stored by the policy checker. On the other hand, this mechanism is free of false positives.

D. Mechanism IV: ElGamal with RSA Accumulators

Instead of prime numbers, an RSA accumulator can be used to store policies. Each ciphertext pair (c_i, c_j) is enumerated by a straightforward linear transformation such as: $e_l = (q-1) \times (i-1) + j$, for $1 < l < k \times \Omega$. The accumulator used by the policy checker is generated by: $Acc(Q) = a^{e_1 e_2 \dots e_{k|P|}} \bmod N$, where $e_1, e_2, \dots e_{|kP|} \in Q$. The corresponding witness for each instance of e_l is generated as: $w_l = a^{e_1 e_2 \dots e_{l-1} e_{l+1} \dots e_{k|P|}} \bmod N$. Verification is done by checking the equality of $w_l^{e_l} = Acc(Q) \bmod (N)$. This solution is both robust against statistical attacks and compact.

TABLE I
COMPARISONS OF PROPOSED MECHANISMS.

	Space		Time
Mechanism	Policy DB (Q)	Packet (u_i)	Policy check
I	$O(m)$	$m \log m$	$O(m^2)$
II	$O(km)$	$\log m$	$O(m)$
III	$O(km \log n)$	$\log m$	$O(m/\log m)$
IV	$O(1)$	$\log(2m)$	$O(\log m)$

E. Space and Time Requirements

We compare the space and time requirements for the different mechanism. For space, we consider the size of the policy database in its protected representation Q as well as the representation of u_i. For time, we consider the computational requirements to perform a policy check. The results are shown in Table I and depend on the number of valid policies, m, and the number of total policies, n, and the number of different encryption keys, k, in Mechanisms II and III.

We can observe that the polynomial method, Mechanism I, is most expensive in space and time. Mechanism IV using ElGamal with RSA accumulator shows the best scaling properties when going to large m.

VI. CONCLUSION

In this paper, we presented several approaches to the privacy preserving policy checking problem in environment of mutual mistrust. Using a practical example of a military network, we show the trust relationships and argue for the need of a trusted third party. Using probabilistic encryption and RSA accumulators, we derive four mechanisms for policy checks that meet each parties trust requirements. Our solution using ElGamal encryption and RSA accumulators shows the best overall performance in terms of space requirements and computational complexity.

REFERENCES

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of the 2003 ACM International Conference on Management of Data (SIGMOD)*, pages 86–97, San Diego, CA, June 2003.

[2] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. Off by default! In *Proc. of Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, College Park, MD, Nov. 2005.

[3] J. Benaloh and M. de Mare. One-way accumulators: a decentralized alternative to digital signatures. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, Lofthus, Norway, 1994.

[4] D. Boneh. The decision Diffie-Hellman problem. In *Proc. of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, Portland, OR, 1998. Springer-Verlag.

[5] R. Brinkman. *Searching in encrypted data*. PhD thesis, University of Twente, Enschede, Netherlands, June 2007.

[6] Committee on National Security Systems, National Security Agency, Ft. Meade, MD. *National Policy Governing the Use of High Assurance Internet Protocol Encryptor (HAIPE) Products*, Feb. 2007. Policy 19.

[7] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. of of CRYPTO 84 on Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, 1985. Springer-Verlag.

[8] R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, May 1996.

[9] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proc. of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, Apr. 2004. Springer Verlag.

[10] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and M. A. A framework for IP based virtual private networks. RFC 2764, Network Working Group, Feb. 2000.

[11] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.

[12] P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, 15(2):24–32, Mar. 2001.

[13] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proc. of the 2002 ACM International Conference on Management of Data (SIGMOD)*, pages 216–227, Madison, Wisconsin, June 2002.

[14] L. Kissner and D. Song. Private and threshold set-intersection. In *Proc. of the 25th Annual International Cryptology Conference (CRYPTO)*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257, Santa Barbara, CA, Aug. 2005. Springer Verlag.

[15] J. Li, N. Li, and R. Xue. Universal accumulators with efficient nonmembership proofs. In *Proc. of the 5th international conference on Applied Cryptography and Network Security (ACNS)*, volume 4521 of *Lecture Notes in Computer Science*, pages 253–269, Zhuhai, China, June 2007.

[16] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Oct. 1996.

[17] U. Meyer, S. Wetzel, and S. Ioannidis. Distributed privacy-preserving policy reconciliation. In *Proc. of IEEE International Conference on Communications (ICC)*, pages 1342–1349, Glasgow, Scotland, June 2007.

[18] J. Voris, S. Ioannidis, S. Wetzel, and U. Meyer. Performance evaluation of privacy-preserving policy reconciliation protocols. In *Proc. of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 221–228, Bologna, Italy, June 2007.

[19] T. Wolf. A credential-based data path architecture for assurable global networking. In *Proc. of the 2007 IEEE Conference on Military Communications (MILCOM)*, Orlando, FL, Oct. 2007.