Conditional Disclosure of Encrypted Whitelists for DDoS Attack Mitigation

Giuseppe Bianchi, Hanieh Rajabi, Alberto Caponi, Giulio Picierro
DIE, University of Rome Tor Vergata
name.surname@uniroma2.it

Abstract—Defensive techniques against Internet-scale attacks can significantly benefit from sharing network security data among different domains. One compelling example, proposed in this paper, is the case of whitelists for DDoS mitigation, where domains broadcast, for each possible DDoS target (\(T\)), the set of legitimate customers (client IP addresses) whose traffic should not be blocked while a DDoS attack is in progress. However, such a fine-grained whitelist sharing approach appears hardly appealing (to say the least) to operators; not only the indiscriminate sharing of customers’ addresses raises privacy concerns, but also it discloses, to competitor domains, business critical information on the identity and activity of customers. In a previous work, we proposed a cryptographic approach called “conditional data sharing”, devised to permit disclosure of cross-domain shared fine-grained organized subsets of network monitoring data, only when a threshold number of domains are ready to reveal their data. In this paper, we cast such technique to a realistic scenario of whitelist sharing for DDoS mitigation, and we significantly extend the underlying cryptographic approach so as to support disclosure not only for threshold-based policies, but for more general (monotone) access structures.

I. INTRODUCTION

Today, most cyberthreats leverage Botnet infrastructures and are unleashed on a worldwide scale, irrespective of the Internet Autonomous Systems’ boundaries. Proper detection of large-scale attacks can hardly rely on the partial single domain’s viewpoint, and purely local mitigation and filtering techniques can be inadequate to thwart their impact.

The research community has long time advocated [1]–[5] the collaborative exchange of security-related data across multiple administrative domain, so as to improve the effectiveness of threat analyses, correlation, and (ultimately) mitigation. At the same time, cross-domain harvesting and exchange of network security data comes along with severe privacy concerns, and is deemed to disclose confidential business information concerning the involved domains, such as network topology, type and amount of customers, offered services, etc. [6]–[8].

We believe that the real world viability of large-scale cross-domain cybersecurity approaches relies on two different, but both make-or-break, properties.

First, business-competing operators need to disclose the very minimum amount of information strictly necessary\(^1\) to the purpose of countering a given large-scale attack. However, it is a fact that real world experience shows basically no exchange of data beyond that strictly mandated by contractual agreements or regulatory policies. Indeed, security benefits emerging from cross-domain data sharing are hard to trade off with the risks of disclosing private or business critical data. And de-anonymization means [10]–[12] have brought about significant skepticism on the viability of massive collection and sharing of anonymized data. We address this issue by cryptographically conditioning the disclosure of data to the actual occurrence of monitoring alerts which signal that a given attack is emerging. Our approach merges the operational simplicity of indiscriminate collection and sharing of data, with the hard guarantee that this data will be unusable (encrypted) unless a set of domains satisfying an arbitrary monotone access control policy, e.g. \((\text{domain}_A \land \text{domain}_B \land (\text{domain}_C \lor \text{domain}_D)))\), will signal an alert.

A second, complementary, property is the need for a scalable operation, not requiring online protocol-based coordination across domains. Indeed, despite the large amount of work carried out in developing cross-domain security-related data sharing protocols such as the IETF RID (now renewed under the IETF MILE initiative), deployment has been so far scarce; whenever used, RID has been mainly employed for asynchronous delivery of incident reports, rather than as full fledged query/response protocol. Moreover, the number of network entities (e.g., web servers or IP addresses) being monitored may be huge, and may significantly vary with time and domains, so that a preliminary (or online) coordination to decide which entities should be jointly tracked appears overwhelming. Our approach permits autonomous operation among domain, the only coordination needed is performed once for all at (global) setup time; each domain can freely decide, and change in time, which network entities should be monitored, and release of (encrypted) data collected for each entity can be performed asynchronously and independently from any other domain. Actually, since in our approach processing and encrypted data export is done on the fly and without the need to keep states, complexity does not depend on the potentially huge amount of entities tracked (only the data export volume depends on this).

\(^1\)Property also stemming from the necessity principle expressed by the European directive [9]: “a data controller should collect and process only the kind and amount of data that are functional and necessary for the specific processing purpose that is pursued”.

Contribution: This paper aims at overcoming two crucial shortcomings of our previous work [13]. First, [13] remained at the pure theoretical level, with little discussion on its applicability to realistic scenarios. In this paper, we instead show how the conditional data sharing ideas presented in our previous work find a (we believe) compelling application as basic building block for an original whitelist-based DDoS mitigation technique, where domains get to disclose (if and only if an attack is detected) lists of “good” IP address to
prevent from filtering (Fig. 1). We believe that this specific DDoS mitigation technique might open the possibility to shift the balance between the risk of disclosing information and the gain from exchanging data in favor of the latter. Indeed, by (conditionally!) exchanging whitelists of legitimate IP address to be prevented from filtering during a DDoS attack, it is in the (business) self-interest of each domain to exchange data which will be disclosed only upon the occurrence of an actual attack, so as to guarantee an acceptable level of service to its customers even under large scale attacks on sites outside the considered domains.

Second, we significantly improve the viability of [13], by substantially extending its cryptographic approach: from the ability to disclose data only if a threshold number of domains signal an alert (a challenge launched in [7]), to the much more practical ability to disclose data only if said alerts come from specific and selected domains satisfying a given (arbitrary) policy. This extension permits to capture the real world fact that domains are different each other: one can believe in the emergence of an attack when signaled by just one specific and highly trusted tier 1 operator, but may require many more (if any) regional operators to signal the same for being convinced.

II. DDoS MITIGATION VIA FINE-GRAINED WHITELISTS

Defensive techniques against Distributed Denial of Service (DDoS) attack ultimately revolve around the effectiveness in distinguishing a relatively small quantity of legitimate traffic from a potentially huge amount of attack traffic [14]–[16]. Since DDoS attacks typically spread across several domains, defences may significantly gain effectiveness by employing forms of cross-domain collaboration (e.g., exchanging information for detection purposes) and coordination (e.g., traceback and distributed filtering).

Most DDoS related works focus on technical means to (collaboratively) detect and filter out the bad traffic. Conversely, only few works focus on the somewhat dual approach of establishing and sharing whitelists of good addresses [17]–[19], not to be filtered (and/or to be prioritized) under attack conditions. Arguably, the flexibility of whitelist-based approaches could be in principle extremely appealing to operators. Each domain could freely and autonomously decide which addresses shall be guaranteed service, at no risk that any of them may be erroneously considered malicious and blocked (by algorithms and firewalls possibly running outside the domain’s boundary, and hence not controllable by the domain itself).

However, as anticipated in the introduction, whitelist based approaches require domains to share sensible information, and thus it’s hard to apply them on a large set of addresses and/or envision sharing on a massive and highly dynamic scale; indeed, [17] restricts the applicability of its whitelist approach to a coarse single subset of “very important IP addresses”, therein called VIP list. The approach presented below, leveraging the cryptographic construction presented in section III, permits to overcome several concerns, as a potentially very large number of fine-grained organized whitelists can be now managed and selectively disclosed only under precisely specified attack conditions.

The choice of addresses to be included in a domain whitelist may rely on domain-specific criteria, possibly including information gathered from private contractual agreements, in addition (or in alternative) to measurements.

A. Proposed Approach

Per-target whitelist: each domain may dynamically set up and update one whitelist for each possible target. The number of targets may be arbitrarily large. Our proposed approach does not depend on how a target is specified (a network address, a service, a location, etc.), but for presentation convenience we will non restrictively assume that targets are web server names. A domain can maintain a whitelist for any target, say www.ieee-globecom.org or www.facebook.com or www.uniroma2.it, irrespective of whether it is hosted inside or outside the domain. The associated whitelist comprises the set of domain customers (IP addresses) whose access to the target server should be guaranteed even in the presence of a DDoS attack involving that specific server.

Whitelist identifiers: since each domain $D_i$ can deploy at most one whitelist for each possible target name $T_k$, a whitelist is uniquely identified by the pair $W_{i,k} = (D_i, T_k)$.

Deploying a target and its associated whitelist: domains do not need to a priori agree on which targets to track. Our only assumption is that targets are globally identified by a same string $T_k$ across all the domains. Indeed, a domain can dynamically deploy whitelists associated to “new” targets, for instance DNS names first seen in a DNS query/response pair. How to construct and update a whitelist associated to a target, is a domain-specific decision. For instance, the example algorithm employed in our experimental setup (section IV) stores, in the whitelist, clients who have (recently) accessed the target server in normal load conditions, whereas accesses during overload periods are not stored (as they could be part of an attack in progress).

Sharing encrypted whitelist: whitelists are shared as encrypted streams of dynamically added IP addresses (or batches of addresses). Following [13], we non restrictively assume that domains rely on a shared repository. Note that such repository does not need to be trusted; it simply plays the role of a shared storage so as to avoid distributing data directly to each participating domain. As shown in Fig. 2, exported data is organized in the repository using as primary index the target name $T_k$ and as secondary index the domain name $D_i$. The addresses contained in a whitelist $W_{i,k}$ are encrypted using an ordinary and fast symmetric cipher (e.g., AES). Each whitelist uses a different key $K_{i,k}$, initially known only by the owner domain $D_i$, and automatically derived from the target name
Indeed, a new deployed web server targets to be a priori specified and/or collaboratively identified. The sharing of whitelists without being necessarily in charge the actual disclosure decision, but a domain can participate to their whitelists is performed if and only if the target set of IP addresses inserted by the participating operators in instance, with 4 domains control policy is a boolean predicate not involving negation; for domains, expressed as an notion in the next paragraph), all the domains involved in the system operation will be able to retrieve, in clear text, all the whitelists associated to the target $T_k$, and will thus be able to instruct their firewalls according (e.g. block all traffic except the whitelisted IP addresses).

Whitelists disclosure: as detailed in section III, the above introduced Suspected Attack Signals, for a same target $T_k$, include cryptographic material which permits the repository to decrypt the content of all the whitelists $W_{i,k}$. Disclosure of the set of IP addresses inserted by the participating operators in their whitelists is performed if and only if the target $T_k$ is signaled as suspicious by a specific combination of the involved domains, expressed as an arbitrary monotone access control policy [21]. We recall that, informally, a monotone access control policy is a boolean predicate not involving negation; for instance, with 4 domains $D_1, \ldots, D_4$ involved, valid policies are $D_1 \land (D_2 \lor D_3 \lor D_4)$, or $D_1 \land D_2 \land D_4$. The latter case remarks that not all domains must be necessarily involved in the actual disclosure decision, but a domain can participate to the sharing of whitelists without being necessarily in charge of contributing to detect a DDoS attack.

Further Remarks: the proposed approach does not require targets to be a priori specified and/or collaboratively identified. Indeed, a new deployed web server $T_k$ implicitly brings about the associated whitelists $W_{i,k}$ irrespective of whether they are actually filled by domains $D_i$ or left empty. Also, no requirement are set on the type and number of targets $T_k$. Indeed, a domain $D_i$ which decides not to track a target $T_k$ (either on purpose or simply because no traffic addressed to $T_k$ travels in the considered domain) will neither send any content for the relevant - implicitly defined - whitelist $W_{i,k}$, nor will ever send an $SAS_{i,k}$.

### III. POLICY-BASED WHITELIST DISCLOSURE

Even if the problem of disclosing the whitelists’ content using a policy-based access structure reminds that addressed by Linear Secret Sharing Schemes [22], it is not a priori obvious which quantity, in our approach, may play the role of a secret to be shared, as each whitelist is encrypted with a different and independent pseudo-random stateless key $K_{i,k}$ computed as per formula (1), and such keys do not have any special structure or interdependency (the per-domain secrets $S_i$ in (1) are all independent each other). Moreover, a crucial requirement which makes the actual solution not straightforward is that disclosure of whitelists for a target $T_k$ must not reveal whitelists associated to any different target $T_k'$. This implies that a secret sharing approach must entail some sort of distinct and independent secrets for each target, which cannot be a priori set up since i) targets are not known in advance and ii) brand new targets may be deployed at runtime. And, finally, for viability reasons, the proposed approach must be fully distributed and must not involve any trusted third party.

As shown in our previous work [13], the above issues were addressed by running a distributed key generation scheme a la Pedersen [23] devised to construct a unique shared master secret $S$ not known by any domain and its associated public key $g^s$, and by reconstructing shares of private keys $H(T_k)^s$ of a Boneh-Franklin’s Identity-Based Encryption (IBE) scheme [24] which uses, as “identities”, the target names $T_k$. In what follows, we extend all this to the more complex case of Linear Secret Sharing Schemes [22] defined by a general access control structure implementing the desired disclosure policy.

#### A. Adversary Model and Security Requirements

Since the entities involved in our scenario are legitimate Internet domains, it seems fair to assume honest-but-curious adversaries. Based on such model, involved domains are expected to regularly follow the protocol rules (i.e. without cheating), but still attempt to gain knowledge. The resulting security requirement is thus straightforward: a set of colluding domains which do not satisfy the given disclosure policy must not be able to gain information about other domains’ whitelists, and cannot deduce nothing about the master secret.

#### B. Setup

In the setup phase, domains obtain shares of a master secret $s$ unknown to any single domain, and compute a public quantity $g^s$ related to such a secret. The secret is suitably distributed such that it can be in principle recovered by any combination of domains which satisfy an initially agreed
Access structure A. This is accomplished by means of a Linear Secret Sharing Scheme (LSSS) built on such access structure. Standard techniques are available to transform an arbitrary monotone policy into an access control matrix, see e.g., [25].

Our starting point consists in enabling parties to share a secret according to the considered access structure A, without relying on any central authority. Let $D_i$, $i \in (1, n)$, be the involved domains, which agree on the following public quantities:

- two large primes $p$ and $q$ such that $q$ divides $p - 1$;
- an $m \times \ell$ access matrix $A$ on domain credentials, representing the specified policy to access the secret;
- a cyclic group $G_p$ of prime order $p$, and a generator $g \in G_p$ for the group. The group $G_p$ is specifically chosen as the domain of a non degenerative bilinear map $e : G_p \times G_p \rightarrow G_T$.

Each party $D_i$ performs the following steps:

- chooses a random secret $\sigma_i \in Z_q$ over the ring of integers modulo prime $q$;
- chooses a random vector $v_i \in Z_p^\ell$ with $\sigma_i$ as first entry;
- for each access matrix row $j$, computes the share $w_{i,j} = A_j \cdot v_i$, and sends it to party associated to the row $j$, using a secure unicast communication;
- computes $g^{\sigma_i} \in G_p$, and broadcasts it to all parties.

Let’s define as master secret the quantity $s = \sum_{i=1}^n \sigma_i$. At the end of these steps, no domain has knowledge of the shared master secret $s$, but each domain is able to compute two crucial quantities:

- the global public key $g^s = \prod_{i=1}^n g^{\sigma_i} \in G_p$;
- the local per party share $x_j = \sum_{i=1}^n w_{i,j} \in Z_q$.

C. Deployment of a new target

Whenever a domain starts to monitor a new target $T_k$ (and build a whitelist associated to it), it must first “deploy” the target, namely deliver to the shared repository suitable (and build a whitelist associated to it), it must first “deploy” the target, namely deliver to the shared repository suitable.

Let’s define as master secret the quantity $s = \sum_{i=1}^n \sigma_i$. At the end of these steps, no domain has knowledge of the shared master secret $s$, but each domain is able to compute two crucial quantities:

- the global public key $g^s = \prod_{i=1}^n g^{\sigma_i} \in G_p$;
- the local per party share $x_j = \sum_{i=1}^n w_{i,j} \in Z_q$.

D. Suspected Attack Signals and Whitelists Reconstruction

Whenever a domain $D_i$, based on its internal monitoring of a target $T_k$, believes that such target may be under DDoS attack, it delivers to the repository a Suspected Attack Signal constructed as the domain $D_i$ share of the private key associated to the target $T_k$, i.e.,

$$SAS_{i,k} = H(T_k)^{s_i} \in G_p$$

When the shared repository (alternatively, the domains if we do not rely on a shared repository but we directly broadcast information among the domains) finds that a sufficient number of shares delivered inside the signals $SAS_{i,k}$ for the target $T_k$ satisfy the access control policy, it determines the coefficients $c_i$ such that

$$\sum_{i \in Q} c_i \cdot A_i = (1, 0, \ldots, 0)$$

where $Q$ is the set of shares released for the target $T_k$, and $A_i$ is the row of the access matrix associated to the share released by domain $D_i$. The IBE private key associated to the target $T_k$ is now trivially recovered by computing

$$H(T_k)^s = \prod_{i \in Q} SAS_{i,k} = \prod_{i \in Q} H(T_k)^{c_i x_i} = H(T_k)^{\sum_{i \in Q} c_i x_i}$$

Once $H(T_k)^s$, the private key associated to $T_k$, is known, all IBE-encrypted keys $K_{*,k}$, thus including the domains which have not detected the occurrence of the attack and sent the relevant SAS signal, can be decrypted, by computing

$$H_2(e(H(T_k)^s, g^s))$$

and recalling, from the properties of bilinear pairings, that $H_2(e(H(T_k)^s, g^s)) = H_2(e(H(T_k)^r, g^r))$, hence, from (2):

$$K_{i,k} \oplus H_2(e(H(T_k)^r, g^r)) \oplus H_2(e(H(T_k)^s, g^s)) = K_{i,k}$$

The keys $K_{i,k}$ are now used to decrypt (and hence concretely share) all the whitelists associated to the target $T_k$.

E. Security Analysis

The system is secure as long as the master secret $s$ remains unknown to all the involved parties and attackers. We shall thus assume that colluding domains do not satisfy the access policy, otherwise it would be trivial for them to derive the secret $s$ and decrypt all possible whitelists.

From [24], it follows that the disclosure of the IBE private key $H(T_k)^s$ associated to a target $T_k$ does not reveal any information about the remaining targets. Hence, the security analysis is straightforward, considering the honest-but-curious adversary model (note that at least for what concerns the part of our approach involving a distributed version of IBE, security results against stronger - byzantine - adversaries are available in [26]). The analysis of our scheme can follow the same arguments provided by Pedersen in [23] (which would be insecure against malicious adversaries, as argued in [27]). The fact that we do not use a threshold-based secret sharing technique, but a more general LSSS does not affect the security of the construction. In fact, consider a colluding subset $Q'$ of honest-but-curious parties interested in the unauthorized disclosure of the master secret $s$. These cheating domains may directly combine their shares $x_i$, so that they can recover.

\[^{46}\text{In general, depending on the access structure, } m \text{ can be greater than the number of domains } n \text{ (i.e. a domain may require to be given multiple shares for satisfying the access policy); with abuse of notation, to make the presentation compact, we non restrictively present follow for the special case } m = n.\]
a quantity $s' = \sum_{i \in Q} c'_i x_i$. However, under the assumption that colluding domains do not satisfy the access control policy, $s' = s + \text{random}$, hence they cannot learn any information about the master secret $s$.

IV. IMPLEMENTATION

In order to gain insights on the proposed operation, we implemented the system in software, using C++ as programming language. XML has been used to structure and transport exported data across domains. Cryptographic primitives were implemented using the OpenSSL library and the Pairing Based Cryptography (PBC) library. As shown in Fig. 3, a participating domain analyzes and exports traffic using a two-stage traffic flow architecture comprising of a first stage, the traffic analysis/monitoring module, and a second stage, a crypto module whose task is to perform all the cryptographic operations associated to the data export functionalities. This architecture permits to decouple the monitoring part, devised to analyze traffic, intercept new targets, and trigger suspected attack signals, from the cryptographic primitives and operations.

DDoS Detection: Since the detection of traffic anomalies is not central in this paper, and since our solution is agnostic to the specific algorithms used to detect whether a target may be under attack, we just sketch a simple approach which can be used as starting point for more comprehensive and effective traffic monitoring and DDoS attacks detection.

Under the working assumption that every web server is a possible target, a monitoring probe captures HTTP traffic, namely packets matching the rule (ip.proto==TCP) && dst.port==80), and, for each destination server, it tracks the number of distinct IP sources in different time windows. The probe sets (measures) a default average load condition, and correspondingly sets two thresholds: an “alert” threshold, greater than the average load, and an even greater “attack” threshold. When the traffic load is below the “alert” threshold, all IP source addresses are considered legitimate clients and are added to the whitelist associated to the server; when traffic grows above the “alert” threshold, new incoming addresses are not included in the whitelist; finally, when traffic grows above the “attack” threshold, a SAS signal is triggered for the target server.

Whitelist sharing: At set up time, the involved domains collaborate once for all to determine the master secret and gather the relevant public key and each domain’s share of the secret according to the access structure agreed, and stores these two quantities in the crypto module illustrated in Fig. 3 along with the domain specific secret $S$, used in (1). Per-target keys are not stored, but generated at each request arrival time using (1). For each request, the monitoring probe extracts the tuple (source IP address, target $T_k$).

A classification module determines whether the target $T_k$ is new (in a given time memory period) or it has been already requested in the past. If the target is old and the traffic load is normal, the crypto module simply encrypts with AES-256 the IP source address requesting the target, and delivers it to the repository. If instead the target is new, the crypto module needs to additionally deliver the IBE encryption of the associated AES key $K_{i,k}$. As shown in the next section, this is by far the most expensive operation of the system as it involves pairings, but it is done only once per new deployed target.

The monitoring probe is further programmed to efficiently (we use on-demand decaying bloom filters [28]) runtime track the average number of distinct requests per target. When the probe signals that this value overflows the “attack” threshold for a target $T_k$, the crypto module computes and delivers the SAS signal containing the IBE private key share $H(T_k)^{\gamma_i}$. Moreover, it marks the target as a candidate DDoS attack and polls the repository for the other domain whitelists (when they will be eventually disclosed) so as to instruct the traffic filters. A summary of the data exported in the various cases is provided in Table I. As export format, we serialize the data in an XML message.

Shared Repository: We further deployed a central physical shared repository as collector of all the exported data. The repository indexes the data according to the target $T_k$ and the domain $D_i$ to distinguish the encrypted whitelists $W_{i,k}$ belonging to different domain identities. Note that the data is (normally) in encrypted form, and the repository does not have the decryption keys, hence the storage is blind. The repository further stores in RAM the received SAS$_i$ signals; for each new signal received, the repository checks whether the disclosure policy is satisfied for the corresponding target $T_k$, in which case it performs the reconstruction of the associated IBE private key using (5) and (6), and once the whitelist keys are recovered, it performs decryption of all the whitelists $W_{i,k}$ associated to the target $T_k$.

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance and scalability of our approach by first assessing the computational time needed to perform each individual cryptographic primitive, and then assessing its application to a real traffic scenario. The performance evaluation is performed on an Intel Xeon X5650 (2.67 GHz, 6 cores) equipped with 16 GB RAM and

![Fig. 3: Architecture of each Domain’s data export system](image-url)

TABLE I: Data exported

<table>
<thead>
<tr>
<th>target $T_k$</th>
<th>Status</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old, New $IP_{i,source}$</td>
<td>Normal</td>
<td>$T_k, E_{T_k}(K_{i,k}), AES(1P_{i,source})$</td>
</tr>
<tr>
<td>Old</td>
<td>Alert</td>
<td>$T_k, AES(1P_{i,source})$</td>
</tr>
<tr>
<td>Old</td>
<td>Attack</td>
<td>$T_k, H(T_k)^{\gamma_i}$</td>
</tr>
</tbody>
</table>
TABLE II: Cryptographic Performance Measurements

<table>
<thead>
<tr>
<th>Function</th>
<th>Performance (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBE encryption, eq.2</td>
<td>2.2051</td>
</tr>
<tr>
<td>IBE decryption, eq.6</td>
<td>2.3070</td>
</tr>
<tr>
<td>Symmetric key derivation, eq.1</td>
<td>0.0086</td>
</tr>
<tr>
<td>Key share computation, eq.3</td>
<td>0.3101</td>
</tr>
<tr>
<td>AES-128 encryption</td>
<td>0.00927</td>
</tr>
<tr>
<td>AES-256 decryption</td>
<td>0.01334</td>
</tr>
<tr>
<td>Key Reconstruction, eq.5 (1 share)</td>
<td>0.35697</td>
</tr>
<tr>
<td>Key Reconstruction, eq.5 (2 shares)</td>
<td>0.98011</td>
</tr>
<tr>
<td>Key Reconstruction, eq.5 (3 shares)</td>
<td>1.18959</td>
</tr>
</tbody>
</table>

An Ubuntu Server operating system. Even if our current preliminary implementation has not been specifically optimized for performance and multi-core exploitation, results are very promising and suggest the feasibility of our system in a realistic setting, although we necessarily leave to future work⁵ the assessment of the full operation of the system in a real world multi-domain scenario under DDoS attack.

Performance of cryptographic primitives: these are summarized in Table II. Performance have been obtained by averaging 1M executions of a same primitive. Results show, as expected, that the greatest toll comes from the IBE-related computations, especially those involving pairings. Symmetric encryption of whitelists is the least expensive operation, but it may be performed in real time during data export, and it is a very bulky task. In fact, IBE-related primitives are very rarely executed, in principle (i.e. assuming no periodic rekeying) only once for every new discovered target $T_i$. Similarly, key reconstruction functions are also done once for-all, and only in the presence of an actual DDoS attack, so the fact that complexity grows with the number of shares is not a major concern. Note that complexity does not depend on the number of domains involved, but only on the number of shares which “solve” the policy. For example, using the policy $\bigcup_{i \in N} D_i$, only 1 share is required to reconstruct the key, while a policy $\bigcap_{A \in D_A} \bigcap_{B \in D_B} \bigcap_{C \in D_C} \bigcap_{D \in D_D} \bigcap_{E \in D_E}$ requires 3 shares.

Viability of the proposed approach: we first observe that the number of involved domains is not critical; it only affect the initial offline handshake and the (occasional, upon DDoS attack) whitelist reconstruction phase which can stand delays. And even considering policies involving tens of domains, Table II suggests that reconstruction delays remain in the several milliseconds order (the theoretical reconstruction time grows linearly with the number of shares). It follows that the system scalability ultimately depends on whether the functions performed in real-time can cope with real world traffic patterns. To answer this question, we gathered both local traffic traces at our premises (Tor Vergata University) as well as tier-1 operator traces provided by CAIDA [29], and we replayed them through our data export system of Fig. 3 (using a 10 Gbps ethernet link for CAIDA traces). The effectiveness of our system is limited by (i) the rate of new targets $T_i$ discovered, as each new target mandates a slow IBE encryption, and (ii) the rate of client addresses to be symmetrically encrypted (whitelist updates). Table III shows such rates for the considered traces. Our operation could easily stand all traces, including the more demanding CAIDA ones. Further experiments devised to stress the system with artificial and suitably crafted traces (traces with one new target per packet or a new IP client per target) have shown that our current implementation, despite the still limited optimizations, can already sustain more than 400 whitelist deployments/sec, and almost 100,000 IP address encryptions/sec.

VI. RELATED WORK

Several defence strategies, including rate limiting, ingress and egress filtering, trace back and push back, have been proposed to thwart DDoS attacks [14]–[16]. The impact of attacks may be alleviated by limiting traffic; filtering mechanisms devised to block attack traffic may be very effective, but pose the risk of denying legitimate users. Deploying DDoS defence mechanisms far from the victim is an essential requirement, although not an easy task: push back techniques pose the problem of identifying common traits in DDoS packets, and trace back solutions are not effective against spoofed IP addresses. Change-point detection across multiple domains, aiming to detect a DDoS flood start is another appealing technique [30].

The idea of using whitelist as a mitigation technique of DDoS attack was proposed in [17] and further exploited in [18], [19]. VIP (“very important IP addresses”) lists were defined in [17]; these list include legitimate users (as ascertained from the analysis of user authentication logins) which are granted higher priority while an attack is in progress. Whitelists are devised based on a binding between user ID and source IP address: if whitelists are long-lived, changes of an user’s IP address may lead to false negatives. Collaborative techniques for anomalous detection and DDoS mitigation are very promising [31]–[34], as they permit to share information about the attack behavior and build global-scale evidence. Decentralized information sharing techniques include gossip-type protocols and epidemic spreading algorithms to share attack information composed of target machines’ IP addresses along with the node’s confidence that certain target machines are under attack [35], to share the origin of anomalous flow along with the frequency of the records in their local domain [36], and so on. However, viability of cross-domain collaborative techniques requires to carefully address (and technically comply with) privacy and regulatory concerns [37]. Privacy-preserving data aggregation and event correlation solutions which permit to disclose the blind aggregated data under a set of conditions have been presented in [38], [39].

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented an approach devised to “conditionally” share whitelists, namely list of addressed to
be protected upon a DDoS attack, characterized as follows: i) whitelists are fine-grained organized on a per-target basis, so that disclosure of one whitelist content does reveal only the set of customers accessing that specific target; ii) disclosure of whitelists occurs only under suspected attack conditions; iii) a cooperative DDoS detection process is established so that whitelists are disclosed only if a set of domains (satisfying an arbitrary policy) signal an emerging attack.

We believe that an important feature of our scheme is the lack of explicit coordination: DDoS targets do not need to be a priori decided, and domains operate in a fully asynchronous manner each other, with no explicit (i.e., request/response) interaction. Finally, our approach does not set in principle any limit on the (possibly very large) number of targets and whitelists each domain shall deploy: preliminary performance results also obtained over real traffic traces show that our approach appears adequate to meet the size and characteristics of real world traffic patterns. Obviously, our next step is to seek support by real world operators so as to test this approach in online operational scenarios.

REFERENCES