International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 4 | ISSN : 2456-3307

A Efficient Privacy Based Multi User Computation Using Inter Domain Routing Decision

Altamash Ali[1] , Md Ateeq Ur Rahman[2]

[1]M. Tech Scholar, Department of CSE, Shadan College of Engineering & Technology, Hyderabad, Telangana, India

[2]Professor, Department of Computer Science & Engineering, Shadan College of Engineering & Technology, Hyderabad, Telangana, India

ABSTRACT

In this paper, we tend to show a network will allow its peers to verify variety of nontrivial properties of its inter domain routing selections while not revealing any further information. If all the properties hold, the peers learn nothing beyond what the inter domain routing protocol already reveals; if a property doesn’t hold, a minimum of one peer will notice this and prove the violation. We tend to gift SPIDeR, a sensible system that applies this approach to the Border entree Protocol, and we report results from AN experimental analysis to demonstrate that SPIDeR encompasses a cheap overhead. Some aspects could also be unconcealed to neighbours, enclosed in a very route written record, or exposed indirectly via glass services, however we tend to cannot expect network operators to conform to use any system that reveals even a lot of their private info. Existing work has shown that it’s attainable to make deductions concerning that autonomous systems area unit connected, and even concerning some aspects of policy however these inferences have restricted accuracy and require extended effort to hold out, creating them unsuitable for substantiate routing selections.

Keywords : Collaborative Verification, Inter Domain Routing, Privacy, Security

I. INTRODUCTION

In inter domain routing; there is an inherent tension between verifiability and privacy: both properties are desirable, but they seem contradictory. Communicating networks have expectations about one another's routing decisions, but they are stymied from verifying these expectations because routing configurations are usually kept confidential. Routing promises. Inter domain routing policies are routinely governed by formal agreements, such as peering and transit contracts, and the correct implementation of these policies is vital for allowing networks to achieve other contractual goals, such as maintaining traffic ratios. In some cases, such as „partial transit‟ relationships, the desired policy can be complex, placing additional cost on the implementers.

II. METHODS AND MATERIAL

2. EXISTING AND PROPOSED SYSTEMS

2.1 Existing System

Secure Inter-domain routing protocols can be verified and validated properties about individual routes for existing system, can be corresponded for real network path. It is often to use a highly complex and highly untraceable properties relating to the procedure which can decide to find the correct route – for example, whether the chosen route was the best one available, or whether it was consistent with the network's peering agreements.

2.2 Proposed System

We can show whether a network can allow its peers to verify a number of significant properties of its inter domain routing decisions without revealing any additional information and its details. If all the properties are on hold, the peers can learn nothing beyond what the inter domain routing protocol already reveals; if some property does not hold, at least one peer can detect this and prove there are some violation. We present a SPIDeR, a practical system that applies this approach to the Border Gateway Protocol, and we report results from an experimental evaluation to demonstrate that SPIDeR has a reasonable overhead.

3. MODULES

A. User Interface Design

B. Data Upload

C. Key Generate & File Sharing

D. Key Request to Data Owner

E. Data Share In Inter Domain

A. User Interface Design

This is the first module of our project. The important role for the user is to move login window to data owner window. This module has created for the security purpose. In this login page we have to enter login user id and password. It will check username and password is match or not (valid user id and valid password). If we enter any invalid username or password we can't enter into login window to user window it will shows error message. So we are prohibiting the network for intruder entering into our login window to new window. It will provide a highly best security for this project. So server contain user id and password server also check the authentication of the user. It well improves the security and preventing from unauthorized data owner enters into the network. In our project we are using SWING for creating design. Here we validate the login user and server authentication.

B. Data Upload

This module is used to help the user to uploading the files. At the time of login, the user could be a valid user means only they allowed uploading their files.

C. Key Generate & File Sharing

In this module is used to help the Group member to encrypt the files and check their file is in safe also providing protection.

Key Generation is the process for generating keys to our files. That key will have to be a unique for every group member while at the time of receives.

D. Key Request to Data Owner

The file is only view format so the file is share and download purpose in Request send to the data owner, the data owner is check the request and user was authorized person so data owner response and key provide to the user.

E. Data Share in Inter Domain

The key was provide to the data owner the user is get the owner ship So user was share the file and download the file.

4. EVALUATION

Next, we report results from an experimental evaluation of SPIDeR. Our goal is to answer two high-level questions: 1) is SPIDeR practical?, and 2) how expensive is SPIDeR? To provide a baseline for comparisons, we aligned our experiments with those from the Net Review paper. Net Review is a good baseline for SPIDeR because it can also verified and promises about inter domain routing policies, and can also be deployed as a companion protocol to Border Gateway Protocol. However, Net Review requires A Ses to disclose a lot of sensitive information, whereas SPIDeR is designed to provide strong privacy guarantees.

4.1 Prototype Implementation

For this experiments, we can built a proof-of-concept implementation of SPIDeR, including a recorder, a proof generator, and a checker. For the recorder, we reused some code from Net Review, specifically the component for mirroring BGP routing state from existing routers and

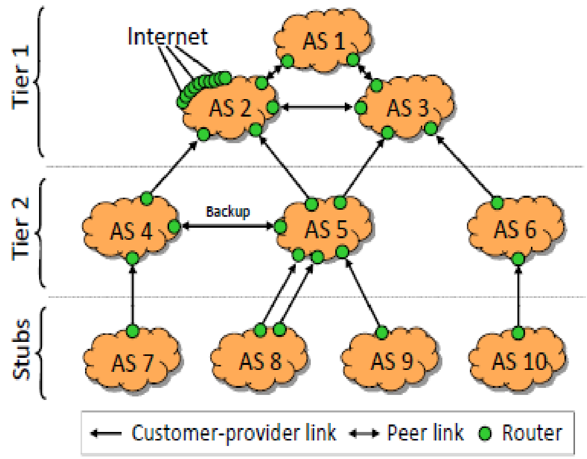


Figure 1. AS topology for our experiments a Route Views trace is injected at AS 2

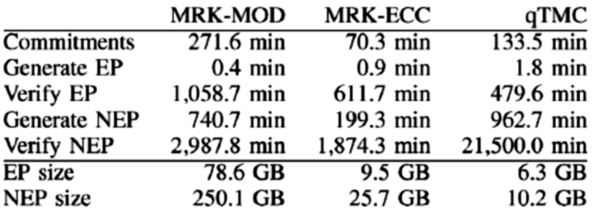
The component for maintaining a tamper-evident message log with signatures and acknowledgments (but not the code for auditing, which is different in SPIDeR). We added a code for the MTT and for a generating commitments; the proof generator and checker are written from scratch. Overall, we added or changed 8,012 lines of C++ code.

We chose RSA-1024 signatures and the SHA-512 hash function, but we use only the first 20 bytes of each digest to save space. The CSPRNG is implemented by encrypting sequences of zeroes with RC4, discarding the first 3,072 bytes to mitigate known weaknesses in RC4. Our recorder implementation uses such separate threads for handling messages and for generating commitments; this prevents the message handler from blocking while MTTs are being labeled. The number c of commitment threads can be varied to take advantage of multiple cores; when $c > 1$, we break the MTT into sub trees that are each labeled completely by one of the threads.

4.2 Methodology and experimental setup

Our result can be estimate the cost a typical Internet AS would incur by running SPIDeR. Since it was not feasible to replicate the Internet's entire AS topology in our lab, we decided to set up a small, synthetic topology (shown in Fig.1) using 36 Quagga BGP daemons in 10 A Ses. However, as, we injected BGP messages from a Route Views trace into one of the A Ses. Thus, the conditions in our synthetic topology were approximately as if the A Ses had been a part of the global Internet: the routing tables contained routes to every reachable IP prefix, and the number and the arrival pattern of the BGP UPDATEs were similar to the conditions at the Route Views collection point.

Specifically, we used a 15-minute Route Views trace that was collected by a Zebra router at Equinix in Ashburn, VA, on January 18, 2012 at 10am. This trace contains 38,696 BGP messages, and the corresponding RIB snapshot contains 391,028 distinct IP prefixes. In our experiment, we first populated the routing tables by slowly announcing the prefixes from the snapshot over a period of 30 minutes; then we replayed the 15-minute message trace. We refer to the first phase as the setup period and to the second phase as the replay period. Unless otherwise specified, we report data that was collected during the replay period, and we focus on the AS in the middle (AS 5).



	MRK-MOD	MRK-ECC	qTMC
Commitments	271.6 min	70.3 min	133.5 min
Generate EP	0.4 min	0.9 min	1.8 min
Verify EP	1,058.7 min	611.7 min	479.6 min
Generate NEP	740.7 min	199.3 min	962.7 min
Verify NEP	2,987.8 min	1,874.3 min	21,500.0 min
EP size	78.6 GB	9.5 GB	6.3 GB
NEP size	250.1 GB	25.7 GB	10.2 GB

Figure 2. Cost of commitments, existence proofs (EP), and non-existence proofs (NEP) with zero-knowledge sets instead of MTTs. Values shown are for AS 5. Note that we used a single core for this experiment; for a fair comparison to SPIDeR, the runtimes should be divided by c=3.

Each AS was configured with a simple routing policy based on Gao-Rexford, defined 50 indifference classes based on the number of hops, and promised to choose the shortest route to all prefixes in the BGP routing Fig.2 table. These simple choices are sufficient for measuring overhead because the cost depends mostly on the size of the MTT and thus on the number of prefixes and indifference classes. Recall from Section 3 that only very few A Ses support more than five local-pref classes, and consider that promises could be made for a specific set of prefixes ("I will give you my shortest route to Google"); hence, 50 classes and all prefixes are both conservative choices.

We ran our experiments on a cluster of 11 machines that were connected by a 1 Gb Ethernet network. Each machine had a 2.4 GHz Intel X3220 CPU with four cores and 4 GB RAM, and ran Fedora Core 10 (Linux 2.6.27.41). For the BGP daemon, we used Quagga 0.99.20 with a 100-line patch that enables the daemon to bind to a specific IP. Commitments were generated every 60 seconds. Unless otherwise specified, we used c = 3 cores for commitments and the fourth core for message handling.

4.3 Micro-benchmarks

We first ran a number of micro-benchmarks to measure the size of a typical MTT, and the time needed to generate and verify proofs.

MTT size: The MTT from AS 5's last commitment in the experiment contains 22,333,767 nodes, including 389,653 prefix nodes, 950,372 inner nodes, 1,511,092 dummy nodes, and 19,482,650 bit nodes. This is expected because there is one prefix node for each IP prefix that is reachable at that point in the trace, and each prefix node has 50 bit nodes. In total, these nodes required about 137.5 MB of memory.

Labeling time: With c = 3 cores, computing the label of this MTT's root node took 13.4 seconds. This seems unproblematic because the computation is done by the recorder (and not by a border router!) and because it is asynchronous, i.e., does not block BGP. Thus, an AS could use our implementation to make a commitment every 15 seconds and catch any promise violations that last at least that long. For comparison, the same computation took 38.8 seconds with only c = 1 core, so the speed-up for c = 3 is 2.9. This is expected because MTT labeling is highly scalable. Because of this, shorter intervals could be achieved by adding more cores, or even additional machines.

Proof generation and proof size: When verification is triggered, the proof generator must reconstruct the MTT at the time of the commitment and then generate a set of bit proofs for each neighbor. For AS 5's last commitment, it took 13.4 s to reconstruct the MTT and 70.2 s to generate the proofs for the five neighbors. The average size of a proof was 449 MB. As a rough approximation, each bit proof with k indifference classes contributes k hashes, or 20.k bytes, plus potentially some hashes of dummy nodes.

For comparison, we also generated the proofs for an alternative promise about just one prefix: "I will give you the shortest route to Google." This took only 0.431 s to generate (after MTT reconstruction), and the corresponding proofs were 2.1 KB for the producers and 2.1 KB for the consumers. If any AS wants to make a declaration that very many prefixes and a proof size is a concern, its neighbors could trigger verification for smaller sub trees, e.g., all prefixes in 32.0.0.0/8.

Proof checking: With c = 1 core, verification of a single proof takes 27 s on average; we observed times from 8.6 s to 40 s. As a first step, the checker needs to rebuild the part of the MTT that is included in the proof and re-label it to verify the commitment; this step took 26 s on average. Then, the checker must verify that all the required bits are present and have the appropriate value; this step accounted for the remaining 1 s.

4.4 Functionality check

Next, we performed a number of sanity checks on our implementation. First, we ran the experiment to completion and then triggered verification; as expected, no broken promises were reported. Next, we re-ran the experiment, injecting different faults into AS 5 that caused its promise to be violated

- **Over aggressive filter:** The AS incorrectly filters out a good route from an upstream AS.
- **Wrongly exporting:** A received route is marked as 'not for export' (with a promise where some routes are worse than the null route) but the AS passes it on.
- **Unauthorized bit proof:** The AS attempts to hide a best route by changing a little bit proof and sent to a downstream AS.

After each run, we triggered verification, and in each case the fault was detected by one of the A Ses. In the first run, the upstream AS raised an alarm because it did not receive a bit proof for the route it had supplied. In the second run, the downstream AS noticed that it had a bit proof for the zero route, which was better than the route it had actually received. In the third run, the downstream AS detected that the proof did not match and the hash value from the commitment. These examples complement the proofs to give us confidence in our implementation.

4.5 Overhead: Computation

The SPIDeR recorder can performs two kinds of operations that are computationally expensive: It signs messages and ACKs, and it also generates and labels a MTT for each commitment. To quantify this overhead, we can used the get usage system call to measure the computation time the recorder's threads spent overall, and we separately instrumented the code to measure the time spent on generating and verifying signatures and on labeling MTTs. We excluded the first and the last minute to avoid startup/shutdown effects.

We found that, during the replay period, the recorder spent 634.5 s of computation time overall. 9.75 s were spent on generating and verifying 3,913 RSA-1024 signatures; note that this is lower than the total number of BGP updates in the trace because, when updates arrive in bursts, the recorder can batch several of them together and sign the entire batch. Generating 13 MTTs required 519 s. All other operations, e.g., BGP RIB maintenance, account for the remaining 105.75 s. Averaged over the entire 13 minutes, a single X3220 core would have been about 81.3% utilized. Since we reuse its messaging code, Net Review would have incurred exactly the same costs, except for the MTT generation; thus, Net Review's CPU utilization would have been about five times lower.

SPIDeR's computational cost increases with the commitment generation rate, and with the number of routing updates that need to be sent (which in turn depends on the number of neighbors), since there are more messages that need signing. For a small AS with five neighbors, like AS 5, the SPIDeR recorder could easily be run on a single commodity workstation. According to CAIDA's topology data, 89% of the current Internet A Ses have five or fewer neighbors.

4.6 Overhead: Bandwidth

SPIDeR can also increases the amount of inter domain routing traffic because all BGP updates need to be re-announced via SPIDeR with additional signatures and acknowledgments. To quantify this overhead, we used tcp dump to capture all BGP packets and all SPIDeR packets that were sent from AS 5 during the replay period. We found that, on average, BGP sent traffic at a rate of 11.8 kbps and SPIDeR at a rate of 32.6 kbps. The relative increase (176%) may seem high, but compared to the amount of traffic A Ses commonly handle, 20.8 kbps is not very much—it is about 2% of a single typical DSL upstream. SPIDeR additionally sends bit proofs to neighboring A Ses for verification; the amount depends on the frequency of verifications and the number of commitments checked which in turn depend on perceived AS needs. Verifying 1% of commitments every minute would result in about 3.0 Mbps of traffic for AS 5.

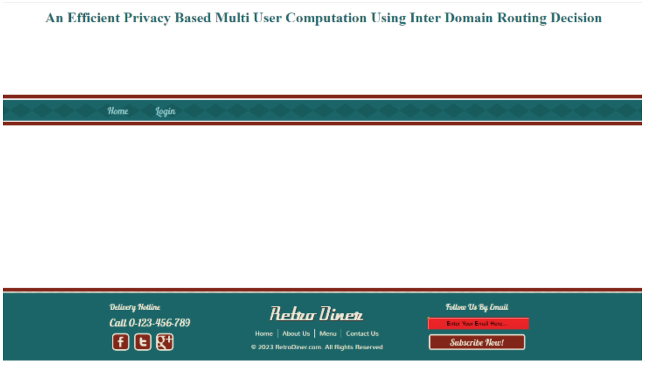
4.7 Overhead: Storage

Each SPIDeR recorder also requires some local storage for the message log, the information is also needed to reconstruct past MTTs, and a number of snapshots of its routing state. To qualify the amount of storage we needed, we can examined the storage of AS 5's recorder after the replay period. We can excluded information that was stored during the setup period.

The log contained 2.95 MB of message data, excluding snapshots; a substantial fraction (24.4%) consisted of cryptographic signatures. Thus, it grew at an average rate of about 232.3 kB per minute. Complete snapshots of the routing state were about 94.1 MB. All of this information would be stored by Net Review as well. There is a only addition in SPIDeR is the MTT-related data, which was comparatively small: each commitment added only 32 bytes to the log. This is because the MTT can be regenerated from the message trace; only the CSPRNG's seed needs to be stored explicitly. Based on these results, we estimate that an AS could keep a year's worth of logs, including one snapshot per day, in 145.7 GB of storage. This data would easily fit onto a commodity hard drive.

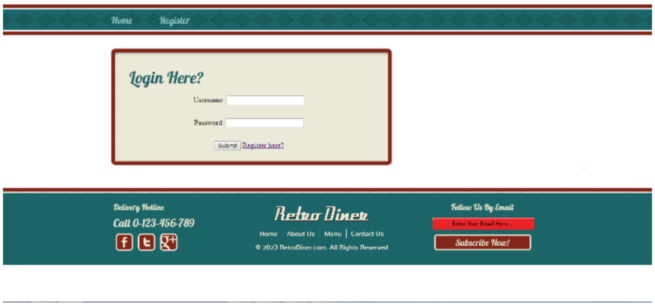
III. RESULTS AND DISCUSSION

IV.



An Efficient Privacy Based Multi User Computation Using Inter Domain Routing Decision

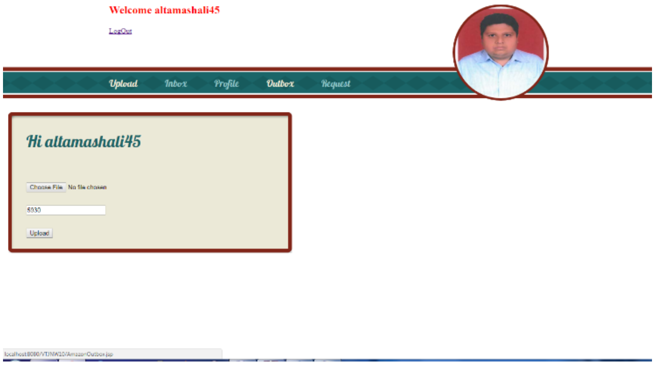
Figure -1: Page-View in the browser



Username:

Password:

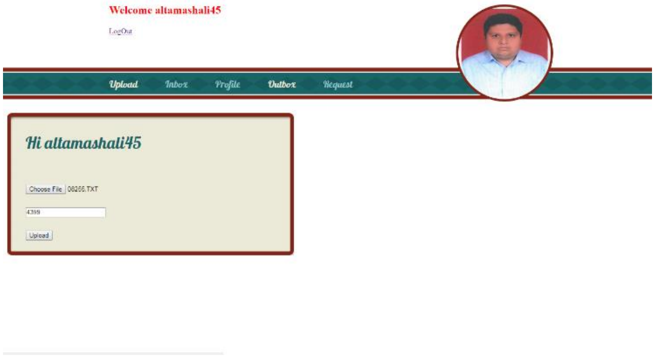
Figure -2: Login Page View in the browser



No file chosen

6930

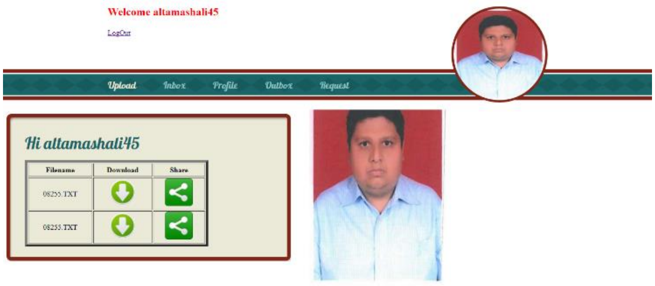
Figure -3: Entering User Name and Password and go to upload button to attach any attachment



08255.TXT

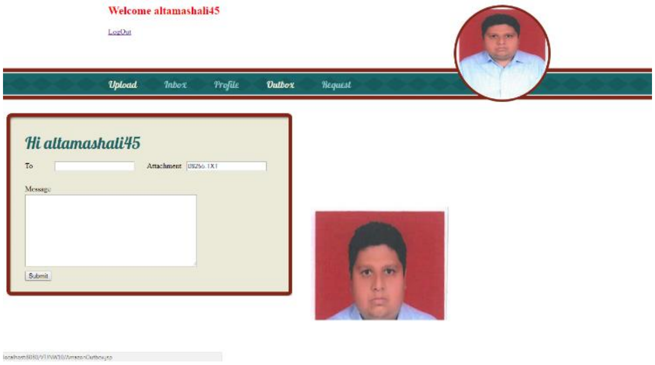
4399

Figure -4: After attaching the attachment the user can upload the data



Filename	Download	Share
08255.TXT	⬇	<
08255.TXT	⬇	<

Figure -5: Uploaded the data can viewed in Profile and the can share to the any user

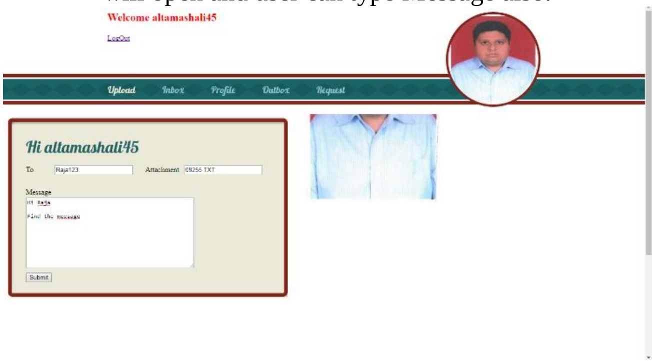


To

Attachment 08256.TXT

Message:

Figure -6: After clicking the share button the messenger box will open and user can type Message also.



To Raja123

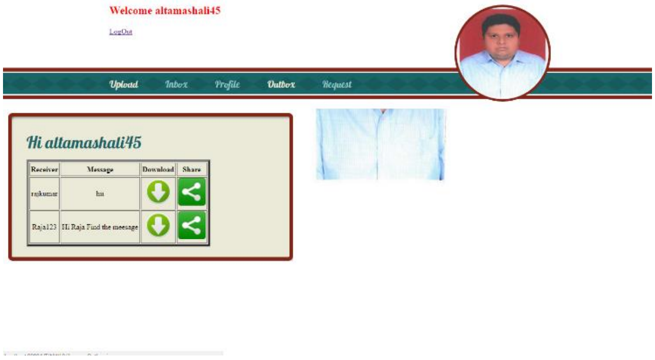
Attachment 08256.TXT

Message

Hi Raja

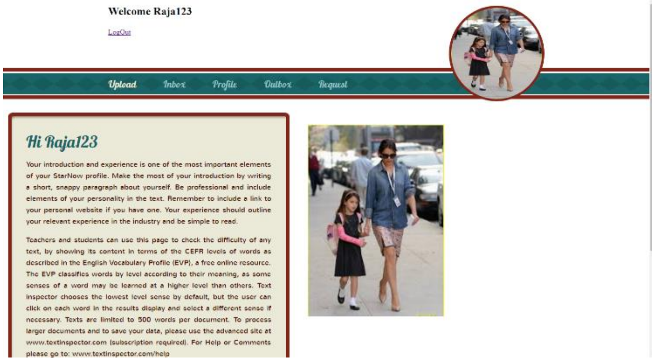
Find the message

Figure -7: After writing the message and to address user can submit his/her request.



Receiver	Message	Download	Share
rajkumar	hai	⬇	<
Raja123	Hi Raja Find the meesage	⬇	<

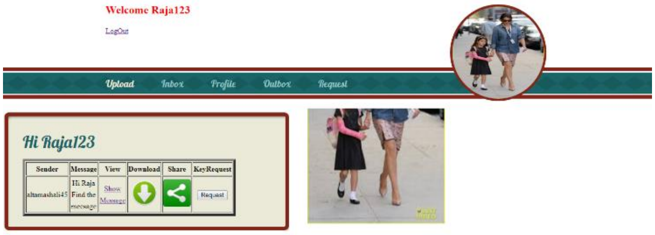
Figure -8: After submit the attachment, by clicking outbox user can also see the sent item.



Your introduction and experience is one of the most important elements of your StarNow profile. Make the most of your introduction by writing a short, snappy paragraph about yourself. Be professional and include elements of your personality in the text. Remember to include a link to your personal website if you have one. Your experience should outline your relevant experience in the industry and be simple to read.

Teachers and students can use this page to check the difficulty of any text, by showing its content in terms of the CEFR levels of words as described in the English Vocabulary Profile (EVP), a free online resource. The EVP classifies words by level according to their meaning, as some senses of a word may be learned at a higher level than others. Text inspector chooses the lowest level sense by default, but the user can click on each word in the results display and select a different sense if necessary. Texts are limited to 500 words per document. To process larger documents and to save your data, please use the advanced site at www.textinspector.com (subscription required). For Help or Comments please go to: www.textinspector.com/help

Figure -9: Now the other user, whom we have sent the attachment open his/her login page.



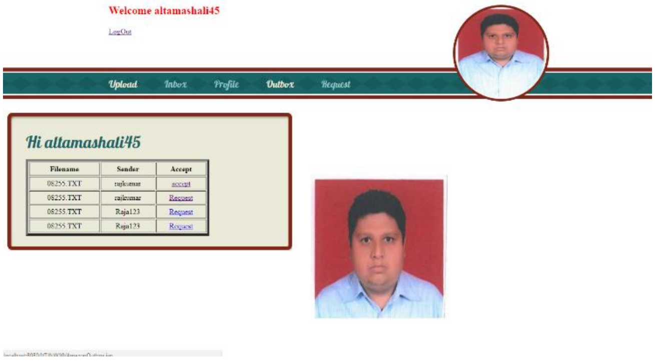
Welcome Raja123

LogOut

Hi Raja123

Sender	Message	View	Download	Share	KeyRequest
altamashali45	Hi Raja Find the message	Show Message			Request

Figure -10: Now the other user, whom should open his/her inbox and the soul user for key, by click request button.



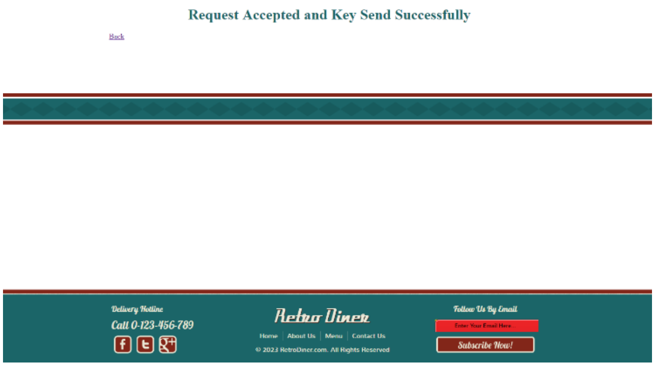
Welcome altamashali45

LogOut

Hi altamashali45

Filename	Sender	Accept
08255.TXT	rajkumar	accept
08255.TXT	rajkumar	Request
08255.TXT	Raja123	Request
08255.TXT	Raja123	Request

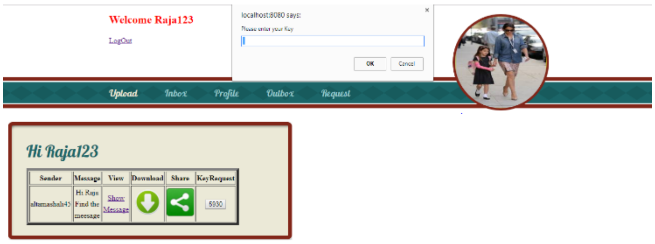
Figure -11: Now the soul user, should accept his/her request by click request button.



Request Accepted and Key Send Successfully

Back

Figure -12: Confirmation message will be showed in web page.



Welcome Raja123

LogOut

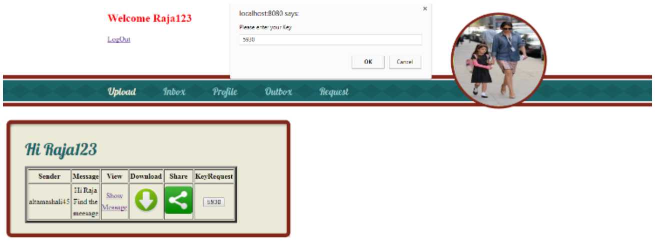
localhost:8080 says:

Please enter your Key

Hi Raja123

Sender	Message	View	Download	Share	KeyRequest
altamashali45	Hi Raja Find the message	Show Message			5930

Figure -13: The other will get the key and to see and download the attachment he/she write the key in page.



Welcome Raja123

LogOut

localhost:8080 says:

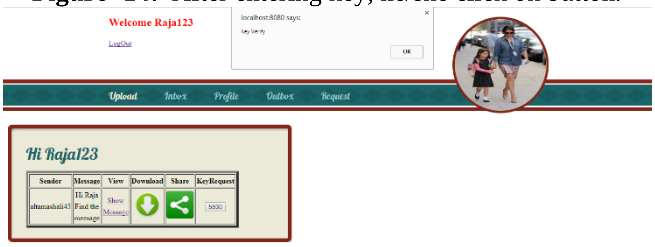
Please enter your Key

5930

Hi Raja123

Sender	Message	View	Download	Share	KeyRequest
altamashali45	Hi Raja Find the message	Show Message			5930

Figure -14: After entering key, he/she click ok button.



Welcome Raja123

LogOut

localhost:8080 says:

Key Verify

Hi Raja123

Sender	Message	View	Download	Share	KeyRequest
altamashali45	Hi Raja Find the message	Show Message			5930

Figure -15: key is verified, the other user can download the attachment.

V. CONCLUSION

This paper has demonstrated that inter domain directing frameworks don't have to settle on a decision amongst undeniable nature and security: it is conceivable to have both. Utilizing our VPref calculation for community oriented confirmation, systems can check various nontrivial guarantees about every others' BGP steering choices without uncovering anything that BGP would not as of now uncover. The outcomes from our assessment of SPIDeR demonstrate that the expenses for the taking an interest system would be sensible. VPref is not BGP-particular and could be connected to other directing conventions, or maybe even to private check undertakings in different areas.

VI. REFERENCES

[1]. Mingchen Zhao, Wenchao Zhou, Alexander J. T. Gurney, Andreas Haeberlen, Micah Sherr, and Boon Thau Loo, "Private and Verifiable Inter domain Routing Decisions", IEEE/ACM Transactions on Networking, Vol. 24, No. 2, April 2016.

[2]. AS Relationships Dataset from CAIDA, Online]. Available: http://www.caida.org/data/active/as-relationships/

[3]. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in Proc. ACM CCS ‘93, Fairfax, VA, USA, 1993.

[4]. O. Bonaventure and B. Quoitin, "Common utilizations of the BGP community attribute," Internet Draft, 2003 Online]. Available: http://tools.ietf.org/html/draft-bonaventure-quoitin-bgp-communities-00

[5]. D. Catalano, M. Di Raimondo, D. Fiore, and M. Messina, "Zero-knowledge sets with short proofs," IEEE Trans. Inf. Theory, vol. 57, no. 4, pp. 2488-2502, Apr. 2011.

[6]. E. Chen and T. Bates, "An application of the BGP community attribute in multi-home routing," in RFC 1998, Aug. 1996 Online]. Available: https://tools.ietf.org/html/rfc1998

[7]. X. Dimitropoulos et al., "AS relationships: Inference and validation," ACM SIGCOMM CCR, no. 1, pp. 29-40, Jan. 2007.

[8]. B. Donnet and O. Bonaventure, "On BGP communities," ACM CCR, vol. 38, no. 2, pp. 55-59, Apr. 2008.

[9]. P. Faratin, D. Clark, P. Gilmore, S. Bauer, A. Berger, and W. Lehr, "Complexity of Internet interconnections: Technology, incentives and implications for policy," presented at the 35th Annu. Telecomm. Policy Research Conf. (TPRC), Arlington, VA, USA, Sep. 2007.

[10]. N. Feamster, Z. M. Mao, and J. Rexford, "Border Guard: Detecting cold potatoes from peers," presented at the 2004 Internet Measurement Conf., IMC ‘04, Taormina, Sicily, Italy, Oct. 2004.

[11]. L. Gao, "On inferring autonomous system relationships in the Internet," IEEE/ACM Trans. Netw., vol. 9, pp. 733-745, Dec. 2001.

[12]. L. Gao and J. Rexford, "Stable Internet routing without global coordination," IEEE/ACM Trans. Netw., vol. 9, no. 6, pp. 681-692, Dec. 2001.