Lightweight Anti-Censorship Online Network for Anonymity and Privacy in Middle Eastern Countries

Tameem Eissa and Gihwan Cho
Division of Electronic and Information Engineering, Chonbuk National University, Republic of Korea

Abstract: *The Onion Router (TOR) online anonymity system is a network of volunteer's nodes that allows Internet users to be anonymous through consecutive encryption tunnels. Nodes are selected according to estimated bandwidth (bnd) values announced by the nodes themselves. Some nodes may announce false values due to a lack of accuracy or hacking intention. Furthermore, a network bottleneck may occur when running TOR in countries with low Internet speed. In this paper, we highlight the censorship challenges that Internet users face when using anti-censorship tools in such countries. We show that the current anti-censorship solutions having limitations when implemented in countries with extensive internet filtering and low Internet speed. In order to overcome such limitations, we propose a new anonymity online solution based on TOR. The network nodes are selected using a trust based system. Most encryption and path selection computation overhead are shifted to our network nodes. We also provide a new encryption framework where the nodes with higher bnd and resources are chosen and verified carefully according to specific metrics. We use an atomic encryption between entry and Exit nodes (Ex) without revealing the secret components of each party. We demonstrate that our solution can provide anonymous browsing in countries with slow internet as well as fewer bottlenecks.*

Keywords: *Anonymity, censorship, TOR, anti-censorship, atomic encryption.*

Receiver August 31, 2012; accepted May 6, 2013; published online August 9, 2015

1.Introduction

The Middle East has experienced massive revolutions demanding the dictator ruling regimes to step down in Tunisia, Libya, Yemen, Egypt, Syria and Iran. This decade has become known as the "Arab spring". In these countries, the regimes have full control over the media. A blackout had been imposed on independent news channels. International reporters are not allowed to enter and report freely. Social networks (Facebook), streaming websites (YouTube), Emails, messaging (Twitter) and VOIP (Skype) were the only available weapons for protestors to report the everyday human right violations committed by the regimes. However, many activists have been arrested by tracking them through the regime censorship systems. The demands for Fine anti-censorship tools are increasing especially for activists and journalists working under such situations. A lasting battle between the regime censorship systems and the anti-censorship techniques and tools was in full swing. The main techniques used for tracking, blocking and monitoring activists are deep packet inspection systems. Telecomix [9] reported the existence of blue coat [13] filtering devices inside Syria and Iran in 2011. Circumventing censorship includes: Bypassing the ISP proxy to obtain access to blocked websites, providing anonymity to hide activists' online identities and encrypting their traffic to secure sensitive information. However, anti-censorship solutions in such countries are facing the following challenges:

- The Internet service in these countries is poor in terms of speed and bandwidth (bnd). For example, the average Internet download speed in Syria between 2009 and 2012 was 0.6 kb, while in Iran it was 0.85 kb. Therefore, implementing anti-censorship tools makes internet service performance even worse. Activists have complained about Internet speed when using services such as The Onion Router (TOR) or VPN.
- Local ISPs censor Internet anonymity services using Deep Packet Inspection (DPI) [11]. Using these systems, ISPs apply packet drops on these services. This causes such services to operate very slowly or may stop functioning altogether.
- Most VPN solutions are blocked by local international gateway and ISPs. The March 2012 report had shown that the VPN PPTP [1] and VPN L2TP [12] were blocked in Syria. While the local censors could drop VPN SSTP [15] packets using deep packet inspection creating a bottleneck in the service.
- TOR [6] is considered the most common anti-censorship tool available in the market, however, users are complaining from delay and internet performance problems when using TOR in countries with slow internet. TOR selects optimum routes according to *bnd* values announced by the nodes themselves. Nodes may make mistakes in

calculating their own *bnd* or announce false values in order to attract more tunnels through them and perform malicious activities.

In this paper, we propose a new TOR based online anonymity solution in an attempt to overcome the previous challenges. Optimum paths between clients and destinations are chosen according to specific metrics, which are evaluated by friendship (fr) based voting system. The main goals are to decrease the *bnd* consumption caused by the cryptography operations on the client side and to enhance the Internet speed affected by these operations as much as possible. To achieve these goals, the number of encryption tunnels on the client side is decreased to one. Furthermore, we shift most of the *bnd* consumption operations from the client to our network to achieve this goal.

Nodes are evaluated not only according to values announced by the node itself but also by validating these values by the node's references. The number of nodes to join the path is flexible according to the user's needs and priorities.

The rest of this paper is structured as follows: section 2 surveys the Internet filtering. Section 3 discusses the most common anti-censorship tools available in the market and their limitations. Section 4 presents our new solution. Section 5 analyses the security of our solution. Section 6 evaluates our solution and finally, Section 7 concludes the paper.

2. Previous Anonymity Solutions

Anonymity is a Greek word meaning nameless. In cyber security, anonymity refers to the state of an internet user being unknown to the public. This includes being invisible to censors or web servers. Anonymity is becoming more popular in online election systems, browsing websites that may track their clients or when using the Internet in anti freedom countries. There are two main methods of anonymity: Message-based anonymity such as email services [8] and flaw based anonymity such as onion routing or peep-to-peer applications [11]. In this section, we highlight two common flaw-based anonymity solutions: TOR and Telex.

2.1. TOR

TOR is an online system providing anonymity and censorship overcoming for internet users [6]. It is widely used by reporters and activists to make their moves invisible from regimes or to gain access into geographical or political blocked websites. TOR uses Onion routing [14] to hide routing information so that, every node in the route has information about only the successor and predecessor hop. Data are transferred in encrypted tunnels. The client should establish several encryption tunnels (usually three tunnels) and send his packets through them. Every node in the route removes one encryption layer using a symmetric key as shown in Figure 1. The figure shows the client establishing three encryption tunnels using three different symmetric keys. Each tunnel is addressed to a specific node since each node can only decrypt one tunnel. Each node in the path should hold a copy of the symmetric key. The symmetric key is established using the Diffie-Hellman protocol. This approach requires high *bnd* at the client side to be able to establish the tunnels and add the encryption layers to his packets before sending them to the TOR network. Every node sends a report about its address, *bnd* and availability (*av*) to the TOR directory. TOR nodes are chosen by TOR servers according to these reports. TOR has been spread widely in the last decade. In 2012, TOR reported 2900 TOR nodes and 1000 TOR bridges [10]. Many governments applied intensive deep packet inspection to block or drop TOR packets and block TOR nodes and bridges. However, TOR always attempts to overcome this censorship using new TOR nodes and bridges or using other techniques such as OBFSPROXY [17].

When choosing routes from the source to a destination, nodes with a higher *bnd* are preferred. However, the node's *bnd* is measured by the node itself. As a result, it is not easy to verify this value. An attacker may make use of this point to broadcast false *bnd* values in order to attract more tunnels through his node. The more encryption tunnels attracted, the greater the chance to reach the first node Entry node (En) and the last node Exit node (Ex). This threatens the anonymity provided by TOR.

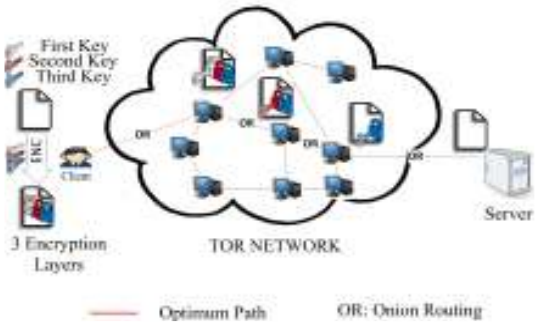

Figure 1. TOR network structure.

2.2. Telex

Telex is a new anonymity tool, which provides the ability to resist website blockage applied by governments and organizations [7]. Telex stations use cryptography tags to identify Telex packets from other packets by means of a shared secret key. When a client requires access to a blocked web server, it encrypts the packets using the Transport Layer Security (TLS) protocol [5], encapsulates them and sends them to a non-blocked website. The local censors allow these packets to pass through. Telex supposes that there is an ISP on the other side supporting Telex and providing Telex stations, which in turn detects Telex packets using the cryptography tags, decrypts them, extracts the original destination address (the blocked website), and re-routes them to this address. When this station receives the response from the blocked website, it encrypts the response packets using TLS and sends

them to the source which in turn decrypts them to obtain the original response.

Telex assumes that ISPs agree on installing and supporting Telex stations, therefore, the success and performance of this service will be affected by the number of ISPs supporting Telex. Furthermore, Telex concentrates on resisting blocking; their solution does not provide anonymity since user information is visible to ISP and Telex stations.

3. Trust-Based Path Selection For Anonymity and Privacy

In this section we present some preliminaries that we believe to be essential aspects for a reader to understand. Afterwards, we present our solution.

3.1. Preliminaries

3.1.1. Bilinear pairings

Let G_1 be an additive group of order q and G_2 a multiplicative group of the same order. The map $e: G_1 \times G_1 \rightarrow G_2$ is called a bilinear pairing [3], if (and only if) it satisfies the following properties:

- Bilinearity: $e(aP, bQ)=e(P, Q)^{ab}$ for all $P, Q \in G_1$, $a, b \in Z_q$.
- Non-degeneratity. Each element of G_1 is appended to an element S from G_2 such that: $S \neq ID_{G2}$ (the identity element in G_2).
- Computability: $\forall P, Q \in G_1, e(P, Q)$ an be computed efficiently.

This cryptography technique has been used in many recent schemes such as in Identity based SDVS scheme [18].

3.1.2. Bilinear Diffie-Hellman Problem (BDHP)

Let G_1 be an additive group of order q, G_2 be a multiplicative group of the same order, and P the generator of G_1 Define the following bilinear pairing on (G_1, G_2):

$$e: G_1 \times G_2 \rightarrow G_2$$

BDHP is the assumption that the following is difficult: Compute $e(P, P)^{abc}$ given P, aP, bP, cP where $a, b, c \in Z^*_q$.

3.1.3. Atomic Encryption

The atomic encryption had been proposed by Blaze *et al*. [2]. In this Encryption, Alice may ask a third party to modify an encrypted message which has been encrypted before by Alice's key. The third party should be able to re-encrypt the message without obtaining access to the original message (plaintext). All he needs is the re-encryption key (proxy key) provided by Alice. The purpose of this encryption is to enable another user Bob to decrypt this message using his own key.

3.2. Thread Model

We define two types of adversaries:

- Adversary 1: this adversary represents the censor which is installed in the local ISP or the international internet gateway. He has the power to filter, drop and block our client traffic. He is also equipped with advanced Internet filtering devices that can block or apply packet droppings on specific security protocols. This adversary is able to execute passive attacks such as traffic monitoring and analysis. He is also able to perform active attacks such as traffic modification, deletion and generation.
- Adversary 2: this adversary does not have the power of adversary 1. He is located beyond the local ISP or international internet gateway. He is able to perform passive attacks such as capturing fractions of the packets. He is able also to operate one or more of our network nodes (Entry, Medium or *Ex*s).

Since, our paper is directed to those seeking anonymity in an anti-freedom country, we will focus on the Adversary 1 attacks.

3.3. Our Solution

Our system approach is close to TOR in terms of network design and routing infrastructure. As previously mentioned, TOR requires the client to add at least three encryption layers to his traffic before sending them out to the TOR network. However, this approach causes bottlenecks and reduced internet speed or breakage in countries with low internet *bnd*. In such countries, many standard security protocols are filtered, dropped or blocked. For example, reports in SYRIA showed that local ISP has blocked many security protocols such as IPSEC and L2TP. They also drop TOR traffic when detecting abnormal TLS packets. As a result, we will direct our efforts to not use such blocked protocols in the connections between the client and the EN since this traffic should pass through the local ISP filters. To solve this problem, we allow the client to add one layer of encryption and shift the task of adding the other layers from the client to our network members. The number of TLS tunnels is flexible in our solution and not restricted to three tunnels as in TOR. This number depends on the trustworthy value of the selected path from the entry to the *Ex*. In some cases, the EN may connect directly to the *Ex* without passing through medium nodes if the direct path is more trustworthy than the other available paths. Furthermore, the task of choosing the circuit is shifted from the client to the En. This reduces the bottlenecks on the client side. We assume that all our network nodes are located in a high Internet *bnd* zone. As in TOR, the network consists of entry, *Ex*s and medium nodes. Each node in the selected path only has information about the predecessor and successor nodes thanks to Onion Routing. The sender encapsulates his

original packets in new packets using single TCP/IP tunnel and sends them to our network. Our network should be able to deliver the packets from the sender, add new encryption tunnels without revealing the original packets and forward them to the final destination. Only the *Ex* should have the ability to decrypt these packets and retrieve the original client's packets. When the destination generates the response packets, our network adds encryption tunnels to this packet allowing the client to perform one decryption process to retrieve the original response. In this way, the local ISP is unable to recognize what website the sender is visiting since, his packets are passing through the ISP encrypted. Furthermore, the final destination server is unable to recognize the sender's identity but can only recognize the *Ex*s identity. Our network structure is shown in Figure 2 where the client performs only one encryption process and sends the encrypted packets through our network. The En re-encrypts the packets using atomic encryption. The proxy key is provided by the client using a secret channel. This atomic encryption allows the third party (which is the Ex in our case) to decrypt the packets and retrieve the original data. The encryption calculation cost is distributed between the client, entry and *Ex*s. In our solution, each organization or each group of activists or journalists may establish its own network and choose their own preferred and trusted nodes to be members of this network.

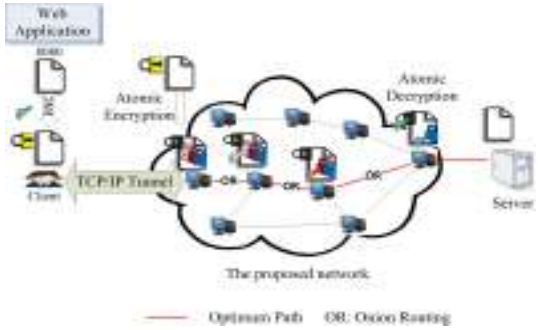

Figure 2. Our proposed solution structure.

3.3.1. System design

- *Setup*: We define the following bilinear pairing e: $G_1 \times G_2 \rightarrow G_2$

  Where G_1 an additive group of order q. q is a large prime and G_2 is a multiplicative group of the same order q.

  We define a random generator P from the group G_1, and a Hash function H: $G_2 \rightarrow \{0, 1\}^*$.

  The main server constructs the nodes key pairs as follow: d_x is node x private key, P_{d_x} is node x public key, d_x™Z_q.

- *Initialization*: The initial nodes are chosen to be the base construction of the network. These nodes should be trusted by our servers.

  Node's score calculation, each node's score is calculated as follows:

$$Node\text{-}score=F(fr, bnd, av, Node)= fr+bnd+av \quad (1)$$

- *Choosing the Optimum Path*: A path X_i is identified by a score, which is calculated as:

$$path\text{-}score[X_i] = \sum_{i=1}^{h} \frac{node\text{-}score[i]}{h} \quad (2)$$

Where h refers to the number of nodes in the path X_i .

The *En* then chooses the path with a maximum path-score value: Optimum-path=max(path-score [X_i]).

Figure 3 shows a scenario of a client who has different Ens and paths with different path-score values [60, 75, 90, 80]. The client chooses <*En, Ex*> as an optimum *En* and *Ex*. The *En* chooses the path with optimum path-score (which in this case is the path with the value '90').

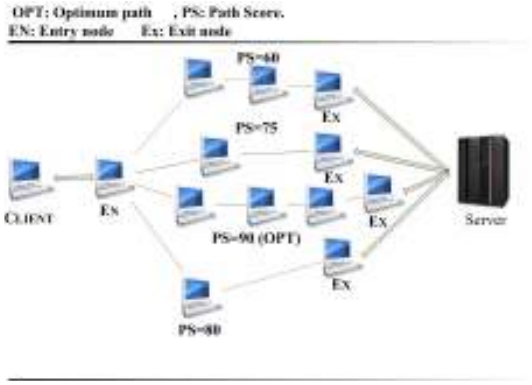

Figure 3. An example of choosing the optimum path from En to Ex.

3.3.2. Client to Destination Communication Workflow

Suppose a client *S* needs to connect to a web server *D*. Suppose "Data" represents the application layer commands and data. For example, the client requests the page www.Example.com, then the "Data" should be: "Begin <website>:80, HTTP GET". We do not want any node in the path except the EN to have any information about the client. Furthermore, we do not want any node in the path except the Ex to know about the final destination. The client constructs the data to be sent in the following form: the first *L* bits represent information concerning the final destination address and "Param" bits. "Param" represents special parameters about the protocol to be used by the sender to connect to the final destination (SSH, TELNET, SSL, TLS …, etc.,). The next bits represent the original data to be sent.

The client chooses the En and Ex according to the metrics mentioned above. The client then picks a random number r from Z_q and encrypts the previous block of Data as follows:

$$Enc(Data', r) = \{C_1 = Data' \text{Å} e(g, P)^r; C_2 = e(r.g, s.P)\} \quad (3)$$

Where s represents the sender's private key. Data represents the new data after adding the Dst IP and "Param" bits. Dst IP represents the website server IP

address. *S* then calculates a proxy key $n_{S\to ex}$ to enable the EN to re-encrypt *Data'* so that only the *Ex* can decrypt them. The proxy key is calculated as:

$$\Pi_{S\to ex} = e(g,P)^{n} \cdot e(r.g, P_{ex}) \quad (4)$$

Where P_{ex} refers to the Ex's public key.

The client then creates a hash value to be used by the Ex for verification as follows: $w=H(e(r.g, P_{ex}))$. The tuple $\langle C_1, C_2, \Pi_{S\to ex}, w, Ex \rangle$ is encapsulated in new packets and sent to the EN through the TCP/IP tunnel.

Upon receiving these packets by the EN, it re-encrypts the data of the received packets using the following proxy atomic encryption:

$$C' = C_2 \oplus \Pi_{S\to ex} = e(g,P)^{r.ex} \quad (5)$$

Then, it constructs a new tuple $\langle C_1, C', \Pi_{S\to ex}, w \rangle$.

These new encrypted data are encapsulated again in new packets. The task of the *EN* is to find the optimum path to the *Ex* using our metric specifications. Then, it uses onion routing to establish TLS tunnels to transfer the encrypted data to the *Ex*. Each node in the selected path only has information about the next and the previous hop. Each medium node receives these packets, forwards them to the next hop in the path according to the onion routing information until it reaches the *Ex*. The *Ex* removes the TLS encryption layers using the Onion Routing protocol, separates the data from the packets and decrypts the encrypted payload using the following proxy decryption:

$$C_1' = C'^{-ex} = e(g,P)^{r}, Dec = C_1' \oplus C_1 = Data' \quad (6)$$

It verifies the integrity of the data by calculating:

$$w_1 = H(C_1'^{ex}) \quad (7)$$

If $w_1 = w$, the data is authenticated.

The *Ex* then extracts the destination server address and the protocol "Param" information from the *L* field, establishes the client request using packet encapsulation and sends the request to the destination.

3.3.3. Destination to Client Communication

When the *Ex* receives the response from the destination, it encrypts the Response Packet Data (RESP) as follows:

$$Enc(RESP, r') = \{C_{b1} = RESP \cdot e(g,P)^{r'}; C_{b2} = e(r'g, ex.P)\} \quad (8)$$

Where *Ex* represents the *Ex*'s private key. It then generates a proxy key that enables the *EN* to re-encrypt the response such that only the sender can decrypt them. The proxy key is calculated as:

$$\Pi_{ex\to s} = [e(g,P)^{r'.ex} \cdot e(r'.g, P_s)] \quad (9)$$

Where P_s represents the sender's public key. The *Ex* calculates a hash value to be used by the sender for verification as follows:

$$w' = H(e(r'.g, P_s)) \quad (10)$$

The tuple $\langle C_{b1}, C_{b2}, \Pi_{S\to ex}, w, En \rangle$ is encapsulated in new packets and sent in TLS tunnels using onion routing protocol and sent back to the *En*. When the *En* receives the response packets, it removes the TLS encryption layers, and re-encrypts the data using the following proxy atomic encryption:

$$C_b' = C_{b2} \oplus \Pi_{ex\to s} = e(g,P)^{r'.s} \quad (11)$$

Then, it constructs a new tuple $\langle C_{b1}, C_b', \Pi_{ex\to s}, w' \rangle$. This tuple is encapsulated in new packets and sent to the client. The client then separates the data from the packets and decrypts the encrypted response using proxy decryption as follows:

$$C'_{b1} = C_b'^{s} = e(g,P)^{r'}, Dec = C'_{b1} \cdot C_{b1} = RESP \quad (12)$$

It verifies the integrity of the data by calculating:

$$w'_1 = H(C'^{s}_{b1})$$

If $w'_1 = w'$, then the response is verified.

3.4. Correctness

We prove the correctness of the cryptography functions in the forward correctness as follows:

$$C_1' = C'^{-ex} = e(g,P)^{r.ex^{-ex}} = e(g,P)^{r}$$

$$C_1' \cdot C_1 = e(g,P)^{r} \cdot Data \cdot e(g,P)^{r} = Data$$

$$w_1 = H(C_1'^{ex}) = H(e(g,P)^{r'ex}) = H(e(g,P)^{r.ex})$$

$$= H(e(r.g, ex.P)) = H(e(r.g, P_{ex})) = w$$

In the same way we prove the correctness of the cryptography operations in the backward connection.

3.5. Nodes Attributes

The *En* chooses the optimum path to the *Ex* according to *fr*, *bnd* and *av* values. *fr* value represents the trust worthiness of the node as evaluated by its friends. Friends can be a service member that knows and trusts this node. It can also be one of our servers. In case *A* has no information about *B* and needs to evaluate it. *A* asks *B* to provide a list of references that may trust *B*. *A* requests for the *fr* values of each reference. This value indicates the amount of trust a reference gives for *A*. Upon receiving feedback from the references, the final *fr* value can be calculated as:

$$fr_{A\to B} = \sum_{i=1}^{n} \frac{fr_{i\to B}}{n} \quad (13)$$

Where *n* is the number of nodes joining the voting.

The minimum acceptable *fr* value is defined as *fr* threshold t_1. As a result, an acceptable node should satisfy $fr_{A\to B} \geq tf$.

The *bnd* value is initially provided by the node itself. Then, this value should be verified by at least *t* neighbors. Where *t* is a threshold representing, the minimum number of nodes as chosen by the client to

validate the node's *bnd*. The *av* value represents how often a node is available as a service relay. This value is assigned by the node itself and according to the node history in interacting as a relay with other nodes.

3.6. Service Joining

Let's say that node "Alice" wants to join as a volunteer in our service, the process goes through the following steps:

- Alice first searches for the closest server available in here area. Alice then sends a validation request to validate the server. The server responds by sending a SrvCrt to Alice. Alice verifies that the SrvCrt is signed by RootCrt. Alice then sends a joining request to this server. The request should include information about her *bnd*, *av* and list of references R_i that may recommend Alice.
- Our server sends a request to Alice's references to verify Alice's trustworthiness and announced metrics. Each reference R_i responds by a message that includes the following values: *fr*, *bnd* value and *av* values.
- The server calculates:

$$F(fr, bnd, av, R_i)=fr\times i_{fr}=bnd\times i_{bnd}+av\times i_{av} \quad (14)$$

Where i_{fr}: the allowed *fr* latency, i_{bnd}: the *bnd* latency, and i_{av}: the *av* latency.

The reference response score is calculated as follows:

$$Reference\text{-}score[Node] = \begin{cases} positive\ if\ \dfrac{\sum_{i=1}^{n} F(fr, bnd, av, R_i)}{n} > t_1 \\ negative\ if\ \dfrac{\sum_{i=1}^{n} F(fr, bnd, av, R_i)}{n} \pounds\ t_1 \end{cases} \quad (15)$$

Where t_1 represents the minimum allowed score.

- The server accepts Alice request if *Reference-score*[*Node*]=*positive*. The server then assigns Alice role which is entry, exit or medium node, according to the value $\frac{\sum_{i=1}^{n} F(fr, bnd, av, R_i)}{n}$. Then, it generates a cryptography key pair to be used for establishing the tunnels. It also generates the specific certificate (*EnCrt, ExCrt or MnCrt*) according to Alice's role. Alice should use this certificate to validate herself to the other nodes. All this information along with the server's details should be sent in a secure channel to Alice.

4. Security Analysis

In this section we prove that our solution provides anonymity and privacy for our clients. We also discuss the security of our scheme under the standard random oracle.

Anonymity is achieved by delegating the task of communication between the destination and our client to the entry, exit and medium nodes. The local ISP won't be able to recognize that these communications are targeted to the destination. The ISP only sees connections to and from *En*'s. Furthermore, the destination server does not know that these communications are originally sent by our client. He only sees requests coming from *Ex*'s.

Same as TOR, our solution encrypts all data transferring between the client, *En*'s, medium nodes and *Ex*'s. However, it does not encrypt the data transferring between the Ex and destinations. If clients need to encrypt these data, it is suggested that the client use the Secure Socket Layer Protocol (SSL protocol) [19] to establish encrypted connections between the client and the destination. SSL is a standard encryption protocol and supported by default by most web servers. It is also supported by all internet browsers. However, technical details about SSL are beyond the scope of this research. In atomic encryption, the sender usually has access to the receiver's private key. However, in our system, we design the proxy key in a way that the *Ex* does not need to reveal his private key to the sender in the forward connection. Furthermore, the *Ex* does not need to reveal its private key to the sender in the backward connection.

4.1. Security Proof In The Random Oracle

We prove that our system is secure in the random oracle model with the assumption that DBDH assumption is difficult.

We define an adversary *A* whose goal is to obtain information about traffic before it reaches the *En* adversary *I*. This includes any entity inside the client's internet zone (this might be ISP or any other party having access to the client traffic before exiting from the international Internet gateway).

We state that our system is secure against Adversary *I* if no adversary *A* has a non-negligible advantage in winning the following game:

At the beginning, the challenger flips a binary coin μ τ(0, 1) out of the adversary's view.

- *Initialization*: We define the following parameters *a*, *b*, *c*, *z* as elements from Z_p as follows:

  A=r; $b=r.s$; $c=r.ex$; $z=a.b.c$

  If $\mu=0$, the challenger sets:

  $$(A, B, C, Z)=(e(g, P)^a, e(g, P)^b, e(g, P)^c, e(g, P)^{abc})$$

  Otherwise,

  $$(A, B, C, Z)=(e(g, P)^a, e(g, P)^b, e(g, P)^c, e(g, P)^z)$$

  The adversary *A* chooses a challenge data (*Data'*) and asks the challenger to encrypt this data. The adversary goal is decrypt the ciphertext encrypted by the challenger. *A* may request the challenger for private keys that can decrypt any data different from the challenge data (*Data'*).

- *Setup*: The challenger runs the setup algorithm and generates the system parameters $<e, G_1, G_2, H, g, P, s, r>$
  The challenger sends the public parameters to A and keeps the secret parameters for himself.
- *Phase* 1: A sends queries for private keys that can decrypt Data different from the challenge data (*Data'*).
- *Challenge*: A chooses two equal lengths Data D_1, D_2 and asks the challenger to encrypt them.

  The challenger randomly chooses b, encrypts D_b and sends the cipher text to A.

$$Enc(D_b, r_b) = \{C_1 = D_b \oplus e(g,P)^{\eta_b}; C_2 = e(r_b.g, c.P)\}$$

Where r_b is a random secret chosen by the challenger. c is the challenger's secret key. The adversary requests the challenger for a proxy key that is used to decrypt the previous ciphertext by any node n' of his choice.

$$\Pi_{C \to n'} = [e(g,P)^{\eta_b c} \oplus e(r_b.g, P_{n'})]$$

Where $P_{n'}$ is the public key of the node n'.

The adversary has access to the following:

$$e(g,P)^{\eta_b}, e(g,P)^{\eta_b c}, e(g,P)^{\eta_b n'}$$

We define the following parameters a, b, d, z elements from Z_p as follows:

$$a=r_b; b=r_b.c; d=r_b.n; z=a.b.d$$

If μ=0 then $Z=e(g, P)^{abd}$, $Enc(D_b, r_b)$ is a valid encryption of D_b.

If μ=1 then $Z=e(g, P)^z$.

Since, z is random; the adversary has no information about the message.

- *Phase* 2: Phase 1 is repeated. Guess Adversary guesses b'; *if* $b=b'$, the adversary wins the game. The probability of winning the game can be defined as:

$$Prb[b = b'] = 0.5.Prb[b = b' \mid \mu = 1] - 0.5 = 0.5 \in \quad (15)$$

Where $\tau\mu$ is the adversary advantage when μ=0.

5. Performance

We use the computation complexity calculation to evaluate the performance of our solution. We focus on the computation complexity on the client device. The main cryptography operations include encrypting client data before sending them to our network, generating the proxy key for the *En* to be able to re-encrypt client's packets and to be decrypted later by the *Ex*, generating the client hash value which is used by the *Ex* to verify the packets integrity, decrypting data received by the client from our network and finally verifying the hash value generated by the *Ex* to assure the response integrity. Table 1 lists the symbols used in the complexity calculation and their meaning. Table 2 shows the complexity of the main operations used in our scheme on the client device. The statistics show that our solution is computationally efficient.

Table 1. List of symbols.

Symbol	Meaning
b	Pairing Operations Cost
m	Scalar Multiplications
e	Cost of Exponentiation Operations
z	Data Packet Size
x	XOR Operation Cost

Table 2. Comparison between our scheme and original TOR.

Operations (Client)	Complexity
Client Encryption	$O(z)+O(2b)+O(2m)+O(e)$
Proxy Key Calculation	$O(2b)+O(2m)+O(e)$
Hash Value Calculation	$O(b)+O(m)$
Data Decryption	$O(m)+O(x)$
Hash Value Verification	$O(m)$

6. Conclusions

In this paper, we discussed the current challenges faced by users seeking anonymity and privacy in countries with extensive censorship and low Internet speed. We proposed a new solution that provides both lightweight and anonymous Internet browsing. The main players in the network, which are entry and Ex's, are chosen according to specific metrics in an attempt to make the service more reliable and trustable. The cryptography overhead is distributed on source, entry and Ex's. The nodes role is chosen according to specific features, which allow the network to be more balanced. Most of the cryptography and path selection overhead is shifted from the client to our network nodes. The system has been proven to be secure against passive and active attacks. Certain suggestions have been made to reduce the network bottlenecks and increase the service performance. This research is believed to be valuable for journalists and activists seeking anonymity during their time spent in Anti-freedom countries.

References

[1] Ancillotti E., Bruno R., and Conti M., "An Efficient Routing Protocol for Point-to-Point Elastic Traffic in Wireless Mesh Networks," *in Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Lucca, Italian, pp. 1-6, 2011.

[2] Blaze M., Bleumer G., and Strauss M., "Divertible Protocols and Atomic Proxy Cryptography," *in Proceedings International Conference on the Theory and Application of Cryptographic Techniques*, pp. 127-144, 1998.

[3] Dan B. and Matthew K., "Identity-Based Encryption from the Weil Pairing- Advances in Cryptology," *in Proceedings of 21st Annual International Cryptology Conference*, California, USA, pp. 213-229, 2001.

[4] Danezis G., Dingledine R., and Mathewson N., "Mixminion: Design of a Type III Anonymous Remailer Protocol," *in Proceedings of Symposium on Security and Privacy*, UK, pp. 2-15, 2003.

[5] Dierks T. and Rescorla E., "The Transport Layer Security (TLS) Protocol," available at: http://tools.ietf.org/html/rfc5246, last visited 2008.

[6] Dingledine R., Mathewson N., and Syverson P., "TOR: The Second Generation Onion Router," *in Proceedings of the 13th Conference on USENIX Security Symposium*, USA, pp. 21-31, 2004.

[7] Eric W, Scott W, Ian G and Halderman A., "Telex: Anticensorship in the Network Infrastructure," available at: https://www.usenix.org/legacy/event/sec11/tech/full_papers/Wustrow.pdf, last visited 2011

[8] Fu X., Zhu Y., Graham B., Bettati R., and Zhao W., "On Flow Marking Attacks in Wireless Anonymous Communication Networks," *in Proceedings of the* 25th IEEE International Conference on Distributed Computing Systems, USA, pp. 493-503, 2005.

[9] Telecomix., available at: http://telecomix.org/, last visited 2015.

[10] Karsten L., "What Fraction of our Bridges are not Reporting Usage Statistics?," available at: https://metrics.torproject.org/, last visited 2015.

[11] Liao M., Luo M., Yang C., Chen C., Wu C., and Chen Y., "Design and Evaluation of Deep Packet Inspection System: A Case Study," *Networks, IET*, vol. 1, no. 1, pp. 2-9, 2012

[12] Mingming H., Qin Z., Kuramoto M., Cho F., and Lunyong Z., "Research and Implementation of Layer Two Tunneling Protocol (L2TP) on Carrier Network," *in Proceedings of the 4th IEEE International Conference on Broadband Network and Multimedia Technology*, Shenzhen, China, pp.80-83, 2011.

[13] PR Newswire, "Blue Coat Delivers High-Performance Web Filtering Using ISS Proventia Web Filter Technology," available at: http://www.prnewswire.com, last visited 2015.

[14] Reed M., Syverson, P., and Goldschlag D., "Anonymous Connections and Onion Routing, Selected Areas in Communications," *IEEE Journal on Selected Areas in Communications*, vol.16, no. 4, pp. 482-494, 1998

[15] Ronald D., John P., Rafal R., and Jonathan Z., *Access Denied: The Practice and Policy of Global Internet Filtering, Information Revolution and Global Politics*, 2008.

[16] Syverson P., Tsudik G., Reed M., and Landwehr C., "Towards an Analysis of Onion Routing Security," available at: http://www.dtic.mil/dtic/tr/fulltext/u2/a465255.pdf, last visited 2001.

[17] TOR Project: OBFSPROXY, available at: https://www.torproject.org/projects/obfsproxy.html, last visited 2015.

[18] Yu H., "Towards Secure Strong Designated Verifier Signature Scheme from Identity-based Systems," *the International Arab Journal of Information Technology*, vol. 11, no. 4, pp. 315-321, 2014.

[19] Weaver A., "Secure Sockets Layer," *The IEEE International Conference on Computer*, Virginia, USA, vol. 39, no. 4, pp. 88-90, 2006.

Tameem Eissa received his BS and MSc degrees in computer engineering from Aleppo University in 2005 and 2007 respectively, and PhD degree in computer science from UTM, Malaysia, in 2012. He is now a Post Doctor research fellow in Chonbuk National University. His research interests include wireless network and cloud computing security.

Gihwan Cho received his BSc and Msc degrees in computer science and statistics from Chonnam National University and Seoul National University in 1985 and 1987 respectively, and PhD degree in computer science from University of Newcastle, England, in 1996. Currently, he is a professor at the Division of Electronic and Information Engineering at Chonbuk National University, Jeonju, S. Korea. His research interests include mobile computing, wireless sensor network, information security, computer communication and wireless security.