
Fang Qi, Zhe Tang, Guojun Wang, and Jie Wu

1 School of Information Science and Engineering, Central South University, Changsha, P. R. China, 410083
2 Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA

e-mail: *csgjwang@mail.csu.edu.cn

Abstract—The primary goal of the secure socket layer protocol (SSL) is to provide confidentiality and data integrity between two communicating entities. Since the most computationally expensive step in the SSL handshake protocol is the server’s RSA decryption, it is introduced that the proposed secret exchange algorithm can be used to speed up SSL session initialization. The optimization strategy, which is based on the constrained model considering the user’s requirements for Quality of Service (QoS), such as security ranking, focuses on the optimal result in different public key size. It is also introduced that the parameter is optimized when integrating user’s requirements for Internet QoS such as the stability of the system and the tolerable response time. Finally, the proposed algorithm is evaluated to be practical and efficient through both analysis and simulation studies.

Index Terms—Quality of Service (QoS), SSL handshake, optimization strategy, security ranking, tolerable response time

I. INTRODUCTION

SSL protects communications by encrypting messages with a secret key negotiated in the SSL handshake protocol [1]. How to offer some Quality of Service (QoS) that may be satisfied with Web users has become a new issue for study.

SSL protocol allows the server and the client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before transmitting and receiving the first byte of data [2]. However, such a protocol needs intensive computational resource due to the cost of public-key operations [3]. Many algorithmic approaches for speeding up SSL’s performance on a web server are presented in some literatures [4-9]. However, these schemes ignore the satisfactoriness of the users’ requirements for QoS such as the stability of the system and tolerable response time.

Being aware of the computational imbalance between clients and server in the SSL handshake protocol, we proposed a secret exchange algorithm to overcome the problem. The starting point of the proposed scheme is a technique due to batch RSA decryption [10]. This paper adapts the certificate mechanism [11] so as to provide SSL setup with unique certificate issued by Certificate Authority (CA). This paper also proposed the constrained model integrating user-perceived quality into secure Web server design [12, 13]. This paper also optimizes the batch size by the constrained model meeting the user’s requirements for quality of service such as security ranking focusing on the optimal result in different public key size. In addition, the proposed scheme in this paper uses the approximate analytical solution of mean response time to optimize the batch size of the server. It is designed for heavily loaded web servers handling many concurrent SSL sessions.

The rest of the paper is organized as follows. Section II describes the secret exchange algorithm in SSL handshake protocol. The proposed constrained model of QoS-aware optimization strategy is presented in Section III. QoS-aware optimization algorithm is presented in Section IV. Section V validates the solutions through both analysis and simulation studies and Section VI concludes the paper.

II. SECRET EXCHANGE ALGORITHM IN SSL HANDSHAKE PROTOCOL

The following Algorithm 1 and Algorithm 2 are secret exchange algorithms of SSL handshake at server end and at client end respectively. When using small public exponent $e_1$ and $e_2$, it is possible to decrypt two cipher texts for approximately the price of one [10,11]. This technology facilitates more favorable load distribution by requiring the clients to perform more work (as part of encryption) and the server to perform commensurately less work, thus resulting in better SSL throughput at the server.

Our unique certificate method is to reuse the message ServerHello.random in the protocol (see Step 2 of Fig.1)[11]. For simplicity, we only show the related processes and the modified information in the standard SSL handshake protocol.

In the standard SSL protocol, each client encrypts a 48-byte pre-master secret using $e_i$ as the encryption exponents, and the server decrypts the cipher text independently so as to get the Pre-master secret. Algorithm 1 obtains the Pre-master secrets from multiple clients and hence improves the performance significantly.
Algorithm 1: Secret exchange algorithm at the server end

1. [Given b distinct and pair wise relatively prime public keys \(e_1, \ldots, e_b\), all sharing a common modulus \(N = pq\), relatively prime to \(\phi(N) = (p-1)(q-1)\). \(n\) is the bit length of the public modulus \(N\) and \(k\) the bit length of the bigger of \(e_i\).]

**Construct a full binary tree \(T_d\) which is called decryption tree with leaves labeled \(e_1, \ldots, e_b\):**

Each node in the decryption tree mainly need conserve two middle values such as \(exponent\) and \(ciphertext\).

2. **Construct message** including \(e_i\) and the information \(e_i'\) about brother’s exponent for each client, where \(1 \leq i \leq b\);

3. **upon receiving** the message including cipher text \(v'_i\) from each client, where \(1 \leq i \leq b\);

4. **Compute** two middle values such as \(exponent\) and \(ciphertext\) at each internal nodes of \(T_d\) repeating this computation recursively. The number of external nodes is equal to \(b-1\).

The computation phase is to generate the product \(v = \prod_{i=1}^{b} v_i^{e_i} \mod N\), where \(e = \prod_{i=1}^{b} e_i\).

For \((i=1\text{ to }b-1)\) -
- \(E_L \leftarrow \text{leftchild}.exponent; E_R \leftarrow \text{rightchild}.exponent;\)
- \(\text{Currentnode}.exponent \leftarrow E_L \times E_R;\)
- \(L \leftarrow \text{leftchild.ciphertext}; R \leftarrow \text{rightchild.ciphertext};\)
- \(\text{Currentnode.ciphertext} \leftarrow \min( L \times R; \text{leftnode.ciphertext}; \text{rightnode.ciphertext}; \text{exponent});\)

(The value of \(v\) and \(e\) is simply the result associated with the root)

5. **Compute** \(m \leftarrow v^{e_i} \mod N = \prod_{i=1}^{b} v_i^{e_i} \mod N\) \(e \leftarrow \prod_{i=1}^{b} e_i\) \(\{m\) is the \(ciphertext\) of root node of \(T_d\). \(e\) is the \(exponent\) of root node of \(T_d\).\}

6. **This Step is to break up the product \(m\) to obtain the plaintexts \(m_i = v_i^{e_i}\) with repeating this computation recursively from the root node.\}

For \((i=1\text{ to }b-1)\) -
- **Compute** \(X\) while ((\(X=0\) mod \(E_L\)) and \((X=1\) mod \(E_R\))) = true)
- \(X_L \leftarrow X / E_L;\)
- \(X_R \leftarrow (X-1) / E_R;\)
- \(m_k \leftarrow m \times (v_k^{X_L} \times v_k^{X_R});\)
- \(m_i \leftarrow m \times m_k;\)

(The values of \(v_k\) and \(v_k\) are simply the results associated with the \(ciphertext\) of node which have stored at step 4.)

**Figure 1. Secret exchange algorithm 1 at the server end**

Algorithm 2: Secret exchange algorithm at the client end

1. **Create** plaintext \(m_i\), \((0 < m_i < N)\); **upon receiving** the message including \(e_i;\)

Compute \(v_i = m_i^{e_i} \mod N\), \(1 \leq i \leq b\)

2. **upon receiving** the message including \(e_i'\) which is the brother’s exponent of \(e_i;\)

Compute \(v'_i = v_i^{'e_i} \mod N;\)

3. **Construct** message including \(v'_i, v_i\) for server.

**Figure 2. Secret exchange algorithm 1 at the client end**

III. CONSTRAINED MODEL OF QoS-AWARE OPTIMIZATION STRATEGY

The optimization strategy which is based on the constrained model considering the user’s requirements for QoS such as security ranking focuses on the optimal result in different public key sizes.

**Lemma 1.** Algorithm 1 can generate the \(b\) decryption results in \(O(\log_b b(\sum_{i=1}^{b} \log_2 e_i) + \log_2 N)\) modular multiplications and \(O(b)\) modular divisions.

**Proof:** According to Step 5 of Algorithm 1, \(m \leftarrow v^{e_i} \mod N = \prod_{i=1}^{b} v_i^{1/e_i} \mod N\), with \(e = \prod_{i=1}^{b} e_i\), the algorithm can get the result in \(O(\log_b N)\) modular multiplications, which is equivalent to one RSA decryption.

Using the full binary tree as a guide, working from the leaves to the root, for constructing the serial number for every exponent of the leaves, the binary length of the serial number is equal to \(O(\log_b b)\). In other words, the depth of the leaves is equal to \(\lceil \log_b b \rceil\).

According to Step 6 of Algorithm 1, the algorithm takes the recursive result from left child and right child, and the result associated with this node is \(m_k \leftarrow m^e (v_k^{e_k} \times v_k^{e_k})\).

The computation phase is to break up the product \(m\) to obtain the plaintexts \(m_i = v_i^{e_i}\), which we wish to decrypt simultaneously.

Note that \(v_i\) and \(v_i\) have already been computed and storied, as the left and right branch values of the root, during the tree based computation of \(m\) at Step 4 of Algorithm 1. By definition \(X\) is the unique solution ((\(X=0\) mod \(E_L\)) and \((X=1\) mod \(E_R\))) = true). Note that \(\log_2 X < \log_2 e\) and \(X_L \leftarrow X / E_L;\) \(X_R \leftarrow (X-1) / E_R;\) we can get \(\log_2 X + \log_2 X < \log_2 X < \log_2 e\) with \(e = \prod_{i=1}^{b} e_i[10]\).

Because the depth of the leaves is equal to \(\lceil \log_b b \rceil\), for every plaintext result \(m_i = v_i^{e_i}\), every node contributes at most \(\lceil \log_b b \rceil\) bits to the appropriate exponents the computation of \(m^e, v_k^{e_k}\) and \(v_k^{e_k}\) recursive result.

Because the binary length of exponent \(e_i\) is \(\lceil \log_2 e_i \rceil\), Step 6 of Algorithm 1 can generate the following \(b\) results in \(O(b\sum_{i=1}^{b} \log_2 e_i)\) modular multiplications or \(O(b \log_2 e)\) modular multiplications with \(e = \prod_{i=1}^{b} e_i\).

To solve for \(m_k \leftarrow m^e (v_k^{x_k} \times v_k^{x_k})\), we divide \(v_k^{x_k} \times v_k^{x_k}\) by \(m^x\), the number of modular divisions required is \(O(b)\).

At all, Algorithm 1 can generate the \(b\) results \(m_1^{e_1} (\mod N), m_2^{e_2} (\mod N), \ldots m_b^{e_b} (\mod N)\).
in $O(\log_2 b\sum_{i=1}^{n} \log_2 c_i + \log_2 N)$ modular multiplications and $O(b)$ modular divisions.

Then Lemma 1 is proved.

**Lemma 2.** Choosing the batch size $b$, which satisfied $2 \leq b \leq \frac{n}{(\log_2 n)^2}$, and choosing the $e_i$ exponents to be polynomial in $n$, we get $O((\log_2 n)^2 + n)$ modular multiplications and $O(\frac{n}{(\log_2 n)^2})$ modular divisions. $n$ is defined as the binary length of modules $N$.

**Proof:** We can easily get $\log_2 N = n$, where $n$ is defined as the binary length of modules $N$.

We can easily get $\frac{n}{(\log_2 n)^2} < n$, because $n$ is a negative number.

Also because the function $\log_2 x$ increases with $x$, we can get $\log_2 \left(\frac{n}{(\log_2 n)^2}\right) < \log_2 N$. Because of choosing the batch size $b$ which satisfies $2 \leq b \leq \frac{n}{(\log_2 n)^2}$, we can derive

$$\log_2 b \leq \log_2 \left(\frac{n}{(\log_2 n)^2}\right)$$

and

$$\log_2 b\sum_{i=1}^{n} \log_2 c_i + \log_2 N \leq \log_2 \left(\frac{n}{(\log_2 n)^2}\right)\sum_{i=1}^{n} \log_2 c_i + \log_2 N$$

Due to Lemma 1, where $\sum_{i=1}^{n} \log_2 c_i = \log_2 e$ and $\log_2 N = n$,

it can be described as

$$O(\log_2 \left(\frac{n}{(\log_2 n)^2}\right)\sum_{i=1}^{n} \log_2 c_i + \log_2 N) < O(\log_2 e(\log_2 n) + n)$$

By choosing the $e_i$ exponents to be polynomial in $n$.

Thus $e < n$, the following equation is derived as

$$O(\log_2 e(\log_2 n) + n) < O((\log_2 n)^2 + n)$$

Then Lemma 2 is proved.

Constrained model considering the user’s requirements for QoS such as security ranking focuses is proposed based on Lemma 2. We optimize the batch size $b$ for a specific modulus size, and obtain better results for smaller batches if the modulus is relatively small. According to Lemma 2, the batch size is optimized as $\frac{n}{(\log_2 n)^2}$ in this constrained model.

**IV. QoS-AWARE OPTIMIZATION ALGORITHM**

Let the decryption time of Algorithm 1 in SSL handshake time be $T_b$. The decryption time in SSL handshake $T_b$ can be estimated as the following [11].

$$3n + n\left(\frac{44b + 3b^2 - 1}{b(3n + n^2)}\right)b_{T_{ba}} = \left(\frac{3n + (44b + 3b^2 - 1)}{b(3n + 1)}\right)b_{T_{ba}}$$

(1)

Since $T_b$ is the majority of the service time, the batching service time of the server $T$ is $T_b$ roughly.

**Lemma 3.** To satisfy the client’s requirement for the stability of the system, the decryption time in SSL handshake $T_b$ is less than the batch size multiplied by the mean Poisson distributed arrival time interval when the time in the Batch Queue Model M/D/1, thus

$$\tau = T_b \leq \frac{b}{\lambda}$$

(2)

**Proof.** Let $X_i (i=1, 2, \ldots)$ be the arrival time interval of two consecutive requests, and $Y$ be the time interval of $b$ consecutive requests. Batch Queue Model M/D/1 has been described in our previous work [11].

If the system achieves the stability when the time $t \to \infty$ for M/D/1 queue model, $T_b < E(Y)$, where $E(Y)$ is the expected value of $Y$. Because the $X_i$ is a random variable with independent identical distribution, the average arrival time interval of $b$ consecutive requests is

$$E(Y) = E(\sum_{i=1}^{b} X_i) = bE(X_i) = b/\lambda$$

(3)

Then Lemma 3 is proved.

**Lemma 4.** In the Batch Queue Model M/D/1, to satisfy the client’s requirement for the stability of the system, thus $T_b < b/2\lambda$.

**Proof.** In the Batch Queue Model M/D/1, the value of $T_q$ is derived following the equation

$$T_q = \frac{1 - e^{-bT}}{1 - e^{-bT} + e^{-bT}} = \frac{e^{-bT} - 1}{e^{-bT} - 1 + e^{bT}} \approx \frac{1}{e^{\frac{bT}{2}}}$$

(4)

Where $T_s = 0.5\tau$, due to Lemma 3, it can be easily described as

$$T_q = \frac{0.5\tau}{1 + \frac{1}{e^{\frac{bT}{2}}}} \leq \frac{1}{(e^{\frac{bT}{2}} - 1)e^{bT}}$$

(5)

It can be easily described as when $b \leq 2$,

$$0.944 = 1 + \frac{1}{(e^{\frac{bT}{2}} - 1)e^{bT}} \leq 1 + \frac{1}{(e^{\frac{bT}{2}} - 1)e^{bT}} < 1$$

(6)

Then the value bound of the upper limit of $T_q$ is estimated as $[0.944 b/2\lambda, b/2\lambda]$. Then Lemma 4 is proved. Tolerable response time (TRT) is the interval between the receipt of the end of transmission of an SSL-based inquiry message and the beginning of the transmission of a response message to the station originating the SSL handshake.
Lemma 5. In the Batch Queue Model M/D/1, to satisfy the client’s requirement for the tolerable response time, RRT<TRT, thus,
\[ \Rightarrow b < 0.4(\lambda \times TRT + 1) \]

Proof. The mean real response time (RRT) is denoted as the sum of \( T_s \), \( T_c \) and the \( T_b \).

In the Batch Queue Model M/D/1, the value bound of the upper limit of \( T_s \) is estimated as \( b/2\lambda \) derived from Lemma 4 (refer to Eq. 6).

The value bound of the upper limit of \( T_s \) is estimated as \( b/\lambda \) derived from Lemma 3 (refer to Eq. 2).

\( T_b \) is the mean time for waiting other client in the same batching which is easily derived that the max value of \( T_b \) is \( (b-1)/\lambda \).

On the other hand, it is supposed that the solution of \( b \) should satisfy the approximate bound, which is derived from Lemma 3, Lemma 4 and described as the following equation:
\[ RRT = T_s + T_c + T_b < \frac{b}{2\lambda} + \frac{(b-1)}{\lambda} < \frac{b}{\lambda} < TRT \]
\[ \Rightarrow b < 0.4(\lambda \times TRT + 1) \]

Then Lemma 5 is proved.

Algorithm 3: QoS-aware optimization algorithm

Input: \( \lambda \), TRT, PKS
Output: Optimal \( b \), \( T_{b \_real} \), Speedup, Speedup\_real

1. Compute the initial value of \( b \)
   \( initial \_b \leftarrow \text{int}(0.4(\lambda \times TRT + 1)) \); (refer to Eq. 7)

2. \( n \leftarrow \text{PKS} \); \( estimate \_b \leftarrow \left\lfloor \frac{n}{(\log, n)^2} \right\rfloor \)

3. If \((estimate \_b < initial \_b)\)
   Then \( initial \_b \leftarrow estimate \_b \);

   Successfind \( \leftarrow \text{false}; \)

4. If \((initial \_b < 1)\) then conventional\_RSA\_decryption();
   return;

5. \( b \leftarrow initial \_b \);

6. While \((b!=1)\)
   do \{ \( T_s = \frac{4n + (4n + 3b^3 k - 1)}{b(3n + 1)} \) \( T_m \) (refer to Eq. 1)
   \( \text{If} (T_b \leq b / \lambda) \) then \{ (refer to Eq. 2)
   Optimal \( b \leftarrow b; \) Successfind \( \leftarrow \text{true}; \)
   break; \}
   Elseif \((T_b > b / \lambda) \) then \( b \leftarrow b-1; \}
   
7. If \((initial \_b < 1)\) then conventional\_RSA\_decryption();
   return;

8. Compute \( T_{b \_real} \);

   Speedup \( \leftarrow \left\lfloor \frac{b(3n + 1)}{3n + (4n + 3b^3 k - 1)} \right\rfloor \); (refer to Eq. 1)

9. Compute \( Speedup\_real \leftarrow Trsa / T_{b \_real} \);

Figure 3. QoS-aware optimization algorithm

Combining the user’s requirements for QoS such as security ranking, the stability of the system and tolerable response time, these strategies aim to optimize the parameter \( b \), which means the size of multi-clients for aggregate decryption in Algorithm 1. QoS-aware optimization algorithm is described in Fig. 4, which satisfies these strategies in this paper.

According to Lemma 5, the initial value of \( b \) is estimated at Step 1 of Algorithm 3 with \( TRT \) and \( \lambda \) as input values. According to Lemma 2, \( b \) is estimated at Step 2 with PKS as input value. If condition of optimal batch can not be satisfied, the algorithm has the ability to fall back on conventional\_RSA\_decryption () which means the decryption with plain RSA, which is described at Step 4 and Step 7. The computation of \( T_b \) is performed using Eq. 1 at Step 6. Step 6 sorts \( b \) to satisfy max solution of \( T_b < b / \lambda \) according to Lemma 3 in descending order from the upper limit computing at Step 3 to two.

V. VALIDATION OF ANALYTICAL MODELS AND PERFORMANCE EVALUATION STUDY

A. Validation of analytical models

The analytical results and simulative results are executed on a machine with a Dell Intel Pentium IV processor clocked at 3.20GHz and 1GMB RAM. Specifically, this paper performs the simulation of SSL handshake secret exchange algorithm with very small public exponents, namely e=3, 5, 7, 11, 13, 17, etc. It is assumed that the value (TRT) is equal to 1 second and 8 seconds as examples both in the analytical model and simulation. It is assumed that the value public key size (PKS) is equal to 512, 1024 and 2048 bits length as examples both in the analytical model and simulation.

Table 1 validates the result of Optimal \( b \) described by the constrained model of QoS-aware optimization strategy. It is assumed that the TRT is equal to 1s in analytical model and simulation. As small arrival rates, \( b \) is almost uniformly calculated by our analytical model (Table 1). Since arrival rates are small (i.e., \( \lambda<2 \)), there is very little opportunity to batch, and therefore, the solution of \( b \) is relatively small (Table 1). Even at higher arrival rate, the analytical result and simulation result are very close. The solution of the optimal batch size is increased with \( \lambda \) both in analytic and simulation when \( \lambda < 30 \) (i.e., PKS=1024bits) approximately. Otherwise, the RRT is not increased obviously. The solution of \( b \) is decreased with \( \lambda \) when \( \lambda > 60 \) (i.e., PKS=1024bits) approximately. The solution of \( b \) cannot satisfy the user’s requirements for the stability of the system, in other words, the solution of \( b \) cannot satisfy the conditions according to Lemma 3 when \( \lambda > 37.5 \) approximately (i.e., PKS=2048bits).
Table 1 Optimal batch size in constrained model validation

<table>
<thead>
<tr>
<th>λ/ PKS</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>60</td>
<td>8</td>
<td>6</td>
<td>-</td>
<td>8</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>8</td>
<td>6</td>
<td>-</td>
<td>8</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>8</td>
<td>6</td>
<td>-</td>
<td>8</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>90</td>
<td>6</td>
<td>5</td>
<td>-</td>
<td>6</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
<td>5</td>
<td>-</td>
<td>6</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

But with non-batching system, it becomes unstable when \( \lambda > 1/T_{\text{kas}} = 1/0.16 = 6.25 \) for 2048 bits keys due to the fact that a non-batching system becomes unstable when \( \lambda > \tau \).

B. Performance evaluation

The simulation result of the RSA decryption time \( T_{\text{kas}} \) with larger public exponent, namely \( e=65537 \) is about 16 ms, 32ms and 130ms with public modulus \( N \) is 512 bits length, 1024 bits length, and 2048 bits length respectively, which is tested using reiterative results.

The multi-factor RSA [9] can expect the theoretical speedup of around 2.25 with \( n = pqr \) and 3.38 for \( n = p^q \). Experiments show the real speedup to be around 1.73 and 2.3, respectively. Rebalanced RSA offers the theoretical speedup of 3.6 but the actual speedup is 3.2 for 1024bits keys. Specifically, \( d \) is chosen to be close to \( n \) such that both \( d \mod (p-1) \) and \( d \mod (q-1) \) are small integers [9]. The resulting public exponent \( e \) also becomes close to \( n \), which is much larger than typical values (i.e., \( e = 3, 17, \) or 65537).

It is in fact so large that Microsoft Internet Explorer (IE) cannot accept it; SB (Shacham and Boneh) scheme [7] offers the speedup factor of 2.5 for 1024bits keys. The downside is obvious because that CA’s charge per certificate regardless of whether the certificate is for the same site or not. It also ignores the satisfaction of the user’s requirements for QoS, where the batch size is equal to four.

Table 2 Speedup of decryption time validation

<table>
<thead>
<tr>
<th>( b/\text{PKS} )</th>
<th>( S )</th>
<th>Speedup</th>
<th>Speedup_real</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>1024</td>
<td>2048</td>
<td>512</td>
</tr>
<tr>
<td>2</td>
<td>1.81</td>
<td>1.90</td>
<td>1.95</td>
</tr>
<tr>
<td>4</td>
<td>3.68</td>
<td>3.05</td>
<td>3.42</td>
</tr>
<tr>
<td>6</td>
<td>2.10</td>
<td>2.80</td>
<td>3.82</td>
</tr>
<tr>
<td>8</td>
<td>1.28</td>
<td>2.18</td>
<td>3.47</td>
</tr>
</tbody>
</table>

Our algorithm offers the speedup factor of 2.76 (Table 2) for 1024bits key which is used in SSL handshake protocol frequently. Typically, \( b \) is equal to 6 for optimal performance when \( 60 < \lambda < 80 \) approximately (Table 1). Obviously, our scheme not only achieves better speedup factor and overcomes these disadvantages of the previous schemes. All the methods are backward compatible with standard RSA. Also, all speedup discussed is based on 1024-bit RSA and is relative to the cost of performing plain RSA decryptions.

It is assumed that \( TRT \) is equal to 8 seconds in Figure 4. These figures show that \( RRT \) is almost linear when \( \lambda \) is relatively small. This is due to the fact that \( RRT = T_e + T_r \) (refer to Eq. 7). When \( \lambda \) is relatively small, the main contribution to \( RRT \) is made by \( T_r \) (i.e., \( \lambda = 10, PKS=1024\text{bits} \)). It is evident that the time \( T_r \) is increased linearly with \( b \). \( T_b \) is also increased with \( b \). Therefore, \( RRT \) is also increased with \( b \) when \( \lambda \) is relatively large (i.e., \( \lambda = 80, PKS=1024\text{bits} \)).

A non-batching system becomes unstable when \( \lambda > 1/T_{\text{kas}} = 1/0.032 = 31.25 \) for 1024bits keys. When the non-batching system is stable, the mean response time \( \tau \) can be estimated as Eq. 8 [11]. The mean service time \( \tau \) is deterministic in the non-batching in M/D/1 queue model. Since \( T_{\text{kas}} \) is the majority of the service time, the mean service time \( \tau \) of the server is roughly \( T_{\text{kas}} \).

\[
T = \tau + \tau \cdot (\frac{\tau \lambda}{2(1-\tau \lambda)} + T_{\text{kas}} + T_{\text{kas}}(\frac{T_{\text{kas}} \lambda}{2(1-T_{\text{kas}} \lambda)}))
\]  

Fig. 5 (i.e., \( TRT = 8 \)) illustrates the comparison of the mean response time of our scheme with non-batching. The vertical axis in each graph is the mean response time over the batch size divided by the mean response time with non-batching. The optimal batch size in our scheme is equal to 10 when \( \lambda = 30 \). The speedup of the mean response time is an optimal one that equals to 6.32 approximately. It is clear that with the optimal batch size our scheme has considerable advantages and while costs little.
In conclusion, this paper proposes the secret exchange algorithm in SSL handshake protocol. Combining the user’s requirements for Quality of Service (QoS) such as security ranking, the stability of the system, and the tolerable response time, these strategies aim to optimize the parameter $b$ which means the size of multi-clients for aggregate decryption. The parameter optimization-based SSL handshake is a viable option for secure communications. Currently we mainly investigated this work based on the Internet with high speed client/server computing paradigm. As our future work, we plan to extend it to more scenarios including P2P networks and wireless networks.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant Nos. 90718034 and 60773013, Hunan Provincial Natural Science Foundation of China No. 09JJ4031, the Program for New Century Excellent Talents in University (NCET-06-0686), and the Program for Changjiang Scholars and Innovative Research Team in University under Grant No. IRT0661.

REFERENCES