ECC Based Threshold Decryption Scheme and Its Application in Web Security*

ZHANG Xian-feng, ZHANG Feng, QIN Zhi-guang, LIU Jin-de
(School of Computer Science and Engineering, UESTC Chengdu 610054 China)

Abstract The threshold cryptography provides a new approach to building intrusion tolerance applications. In this paper, a threshold decryption scheme based elliptic curve cryptography is presented. A zero-knowledge test approach based on elliptic curve cryptography is designed. The application of these techniques in Web security is studied. Performance analysis shows that our scheme is characterized by excellent security as well as high efficiency.

Key words intrusion tolerance; elliptic curve cryptography; threshold decryption; web security

Despite all the defenses, some of the attacks will inevitably be successful. We must build intrusion tolerant systems that can operate through these attacks. The threshold cryptography provides new methodology for developing intrusion tolerance system. The research on the threshold cryptography based intrusion tolerance techniques concentrate on threshold encryption, threshold decryption, threshold authentication and threshold signature. Generally, threshold encryption and threshold decryption are based on the cryptography algorithms, which are homomorphic, such as RSA. The homomorphic property is necessary in order to generate shares of the key so that partial cryptograms can be combined into a cryptogram for the correct message. Therefore, nearly all projects studying on threshold cryptography techniques focused on the homomorphic algorithm. Among these, the intrusion tolerance via threshold cryptography (ITTC) project developed by Stanford University is notable, which researched and developed the intrusion tolerance application based on threshold RSA.

Currently, elliptic curve cryptography (ECC) is regarded as an attractive cryptography that can provide greater strength, higher speed and smaller keys than other cryptography systems. Therefore, the research on threshold ECC deserves great attention. However, ECC is not homomorphic. It is difficult to design ECC-based threshold schemes. So, there are not many public published papers or reports on ECC-based threshold schemes, especially on ECC-based threshold encryption or decryption schemes. In this paper, we present an ECC based threshold decryption scheme and a zero-knowledge test approach. And we apply our research results in building intrusion tolerant Web security system.

1 Preliminaries

Throughout this paper, Let $P>3$ be a prime number and let $\text{E}(A,B)$ be an Elliptic Curve group over $\text{GF}(P)$. Let $H$ be a cyclic subgroup of $\text{E}(A,B)$ such that the discrete logarithm problem is intractable over $H$. Let $g$ be a generator for $H$.

1.1 The Menezes-Vanstone Elliptic Curve Cryptosystem

For the system to be described, plaintext could be any pair $(x_1, x_2)$ of integer from $\text{GF}(P)$. The plaintext $X=(x_1, x_2)$ can be or can be not a point on the elliptic curve. The public key for the system consists of the elements $g, h$ of $\text{E}(A,B)$, whereas the private key consists of an integer $d$. Here, $d \in [1, P-1]$ and $h=dg$. 

Received 2003-10-08

* Supported by the Foundation of National 863 Programme of China (No. 2002AA142040)
Encryption:
1) 0 Select a random integer \(k < \mid H \).
2) Compute
\[ y = kg \mod P \] (1)
3) Compute
\[ (c_1, c_2) = kh \mod P \] (2)
4) Compute
\[ s1 = c_1 \times 1 \mod P \text{ and } s2 = cx2 \times 2 \mod P \]

The encrypted form of \(X\) is \((y, s1, s2)\).

Decryption:
1) To decrypt the message, the holder of the private key \(d\) can compute \((c_1, c_2)\) from the first coordinate \(y\) of the encryption triplet:
\[ dy = dkg = k(dg) = (c_1, c_2) \mod P, \text{ namely } \]
\[ dy = (c_1, c_2) \mod P \] (3)
2) Compute
\[ x1 = s1 \times c_1^{-1} \mod P \] (4)
\[ x2 = s2 \times c_2^{-1} \mod P \] (5)

Hence, the ciphertext is decrypted.

In the cryptosystem, The ECC private key consists of the integer \(d\). To decrypt a ciphertext \(X\), one should compute by Eqs.(3)–(5). Without the secret \(d\), it is hard to decrypt the ciphertext, because this is an elliptic curve discrete logarithm problem (ECDLP).

Note that when a plaintext to be encrypted is too long, it should be encoded into several integer pairs, such as \(f\) integer pairs: \((x1_1, x2_1), (x1_2, x2_2), \ldots, (x1_t, x2_t)\). In process of encryption, step 2 and step 3 only need to carry out one time. Then \((c_1, c_2)\) is worked out. Further step 4 need to carry out \(t\) times to generate the ciphertexts, as \((y, s1_1, s2_1), (y, s1_2, s2_2), \ldots, (y, s1_t, s2_t)\). During decryption, step 1 only need to be performed one time. Then \((c_1, c_2)\) is recovered. Further step 2 needs to be performed \(t\) times to generate the plaintext.

1.2  \((t, n)\) Secret Sharing

If a secret \(d\) is broken up into \(n\) shares, and the secret can only be reconstructed from all \(n\) shares, then the secret sharing scheme is called \((n, n)\) secret sharing. To break up secret \(d\) into \(n\) pieces of shares, we can take the idea presented by Frankel[9], to pick random numbers \(d_1, d_2, \ldots, d_n\) in the range \([-P, P]\) so that
\[ d = d_1 + d_2 + \ldots + d_n. \]

In \((n, n)\) sharing, note that an attacker who breaks into any \(n-1\) of the \(n\) servers learns nothing about the private key \(d\). However, if one of the share servers crashes the entire secret is lost forever. So it is not fault-tolerant. For this reason, \((t, n)\) sharing is preferable. Any \(t\) of the \(n\) share servers can be used to apply the key. A practical \((t, n)\) sharing scheme is presented in Ref.[2]. In the \((t, n)\) sharing, secret \(d\) is split into several combinations (coalitions) of \(t\) different random integers. Every \(t\) shares in one combination is store across \(t\) different servers among the \(n\) share servers.

2  ECC Based Threshold Decryption Scheme

Let \(A\) be the sender, \(B\) be the receiver. \(B\) doesn’t keep his private key \(d\), which is stored across \(t\) out of \(n\) share servers. Let one coalition of \(d\) be
\[ d = d_1 + d_2 + \ldots + d_i \] (6)

Assume that the corresponding share servers to this coalition are \(C_1, C_2, \ldots, C_t\). \(B\) keeps \(d_i\) \((1 \leq i \leq t)\).

\(A\) encrypts a plaintext \(M_d\) to a ciphertext \((y, s1, s2)\) according to section 1.1, and sends \((y, s1, s2)\) to \(B\). \(B\) decrypts the ciphertext as follows:

Threshold decryption:
\(B\) broadcasts \(y\) in encryption triplet \((y, s1, s2)\) to \(C_1, C_2, \ldots, C_t\).
1) Each server \(C_i\) \((i \in [1, t])\) computes
\[ Q_i = dy \mod P \] (7)
and sends \(Q_i\) back to \(B\).
2) After getting all \(Q_i\) \((i \in [1, t])\), \(B\) computes
\[ Q = \sum_{j=1}^{t} Q_j = (q_1, q_2) \mod P \] (8)
3) \(B\) computes
\[ B_1 = s_1 q_1^{-1} \mod P \] (9)
\[ B_2 = s_2 q_2^{-1} \mod P \] (10)
The decrypted form of \((y, s1, s2)\) is \((b_1, b_2)\).

Proof
\[ Q = (q_1, q_2) \mod P = (\text{by Eq.(8)}) \]
\[ \sum_{j=1}^{t} Q_j \mod P = (\text{by Eq.(7)}) \]
\[ \sum_{j=1}^{t} d_i y \mod P = (\text{by Eq.(1)}) \]
\[ \sum_{j=1}^{t} d_j kg \mod P = (\text{by Eq.(6)}) \]
\[ (c_1, c_2) = (\text{by Eq.(3)}) \]
Hence, \((b_1, b_2)=(x_1, x_2) \mod P\) (by Eqs.(4), (5), (9), (10))

3 ECC Based Zero-Knowledge Test Approach

In section 2, if one share server \(C_i\) \((1 \leq i \leq t)\) need authenticate that it has a share \(d_i\) to entity \(B\) without divulging any information about \(d_i\), then \(C_i\) can apply zero-knowledge test approach. Here, we present an ECC based zero-knowledge test approach as follows.

1) \(B\) selects two random integers \(a\) and \(b\), \(a \in [0, |H|−1], b \in [0, |H|−1]\).

2) \(B\) computes \((a_1, a_2)=a (dg) \mod P\) and \((b_1, b_2)=bg \mod P\).

3) \(B\) computes \(t_1=a_1b_1 \mod P\) and \(t_2=a_2b_2 \mod P\), and sends \((ag, t_1, t_2)\) to \(C_i\).

4) \(C_i\) receives message \((ag, t_1, t_2)\), computes \(d_i= (r_1, r_2) \mod P\).

5) \(C_i\) computes \(z_1=t_1r_1^{-1} \mod P\) and \(z_2=t_2r_2^{-1} \mod P\), and sends the computation result \((z_1, z_2)\) back to \(B\).

\(B\) verifies \((b_1, b_2)=(z_1, z_2)\). If this is true, \(C_i\) has the share \(d_i\), else \(C_i\) cannot authenticate that it has \(d_i\).

The correctness of the ECC based zero-knowledge test approach can be proved by the principal of Menezes-Vanstone elliptic curve cryptosystem in section 1.1.

4 Application

4.1 Web Security System Architecture

Web-based applications are pervasive in reality. Web security is important for those web-based applications concerning some critical information, such as financial and martial information. To enable secure connections to a Web server, a Web client (user) often connects to the server by SSL (Security Socket Layer). During the negotiation of SSL session key, Web client need to send Certificate Verity message to the server. The Certificate Verity message includes premaster key signed by the Web client’s private key. The premaster key is encrypted by the server’s public key and sent to the server. After the server receives the message, it should decrypt it with its private. An attacker who penetrates the server can expose the private key and can then either masquerade as the server or eavesdrop on connections to the server. So it is critical to make the private key secure.

The architecture of Web security system is in dotted box illustrated in Fig.1. The system components consist of a Web server, \(n\) share servers, an administrator and an intrusion detection system (IDS).

In the system, the Web server receives access requests from a user through communication link, such as Internet. The Web server doesn’t keep its private key, which is broken up into \(t\) shares by \((t, n)\) sharing and stored across \(t\) different share servers out of the \(n\) share servers. The Web server store each \(d_i\)’s \((1 \leq i \leq t)\) public key \(dg\) and some other public information, such as \(dg\), \(g\), \(P\) and \(E(F_p)\).

The IDS analyzes network traffic and the state of the share servers, the Web server and the administrator to report suspected intrusions. Some IDS modules execute on one or more dedicated hardware platforms, while others reside in the share servers, the Web server and the administrator. We don’t describe our IDS capabilities in more detail here.

The administrator generates and refreshes the shares stored on the share servers. When a new private key is generated it is generated in shared form. The shares of the new private key are stored on the share servers. The new public key is sent to the administrator and is saved on the administrator’s machine. Refreshing the shares does not change the private key. It simply generates a new independent sharing of the private key. Based on \((t, n)\) sharing, suppose an attacker obtains \(r-1\) shares of a private key stored on \(r-1\) different servers. Once the administer refreshes the shares of the private key the information...
in the attacker’s hands becomes useless. Refreshing is also used when shares stored on a share server are corrupted or lost. Refreshing the shares causes the uncorrupted servers to generate new valid shares for the corrupted server. Consequently, the system gracefully tolerates corruption or loss of a few shares. The administrator also can shutdown or suspend the servers if necessary, or instruct the servers to take appropriate action if some of the servers have been penetrated.

4.2 Decryption Scheme

After receiving the encrypted information \( (y, s_1, s_2) \) from a user, the Web server will specify \( t \) share servers from the \( n \) share servers. The selection of the \( t \) share servers can be based on the current workload of the \( n \) servers, to balance every server’s load. The selected \( t \) share servers correspond to a coalition. In term of the coalition, each of the \( t \) servers can independently decide which \( d_i \) it should apply. Then system can decrypt the ciphertext by the ECC-based zero-knowledge decryption scheme presented in section 2.

4.3 Test of Share Possession

If an attack penetrates a share server, he maybe tampers the share \( d_i \) stored on the server. The share server also maybe loses \( d_i \) accidentally. At any rate, the server will send wrong undesired computation result back to the Web server. Therefore, it is necessary to design a scheme to test whether one share server holds \( d_i \). Based on the Web server keeps all \( d_i g (1 \leq i \leq t) \), we can apply the ECC based zero-knowledge test approach in section 4 to test it. By the test approach, the share server can authenticate that whether it has \( d_i \), and reveal no information about \( d_i \).

5 Performance Analysis

We will mainly analyze our threshold decryption scheme in this section. Because the ITTC project also study intrusion tolerant Web application based threshold cryptography, we will analyze our scheme’s performance by comparison method.

In ITTC, the decryption is based on threshold RSA. The Web server’s private key \( k \) is also processed by \( (t, n) \) sharing. When the Web server receives the encrypted information \( M \), system will operate as follows: 1) the Web server sends \( M \) to the \( t \) specified share server; 2) each server decides the \( k_i \) \( (1 \leq i \leq t) \) to be applied, and computes \( M^{k_i} \mod N \) \( (N \) is the modulus of RSA), then sends the computation result back to the Web server; 3) after the Web server obtains all the \( t \) computation results, it computes \( \prod_{i=1}^{t} M^{k_i} \mod N \) and gets the plaintext.

5.1 Security

In our scheme, we protect Web server’s private key by \( (t, n) \) sharing. An attacker who breaks into to \( t-1 \) share servers cannot expose the private key. Additionally, The private key \( d \) is never reconstructed at a single location during the process. Therefore, there is no single point of attack at which an attacker can expose critical security information. This security property also belongs to the schemes in ITTC.

During the decryption operation of our scheme, the \( t \) share servers involved in send computed point \( d_i y \mod P \) to the Web server. It is difficult for a hacker to work out \( d_i \) from \( d_i y \), because this is an ECDLP problem. In ITTC scheme, if a hacker wants to compute \( k_i \) from \( M^{k_i} \mod N \) computed by share servers, he has to face intractability of large integer factorization problem. Currently, ECDLP is believed to be harder than the integer factorization problem\(^8\). From this point, our scheme is more secure than the ITTC scheme.

5.2 Efficiency

In both two schemes, there is no communication between the share servers. The only interaction is between the Web server and each of the share servers. However, there are obvious differences from communication complexity and computation complexity between our scheme and the ITTC scheme.

To compare our scheme’s efficiency with the ITTC scheme’s efficiency in reason, we can suppose that the \( g \)’s order be a 160bit prime described in our scheme and RSA based ITTC scheme with a 1024bit modulus \( N \). Because an elliptic curve \( E(F_p) \) with a \( g \in E(F_p) \) whose order is a 160 bit prime offers approximately the same level of security as RSA with a 1 024 bit modulus \( N \). According to Ref.\(^7\), the following useful conclusions can be used: 1) computing one scalar multiplication \((d_i y)\) requires the
equivalent of 1200 field multiplications; 2) an elliptic curve addition or doubling requires 1 field inversion and 2 field multiplications; 3) the time to perform a field inversion is equivalent to that of 3 field multiplications; 4) a modular multiplication modulo \( n \) takes about 41 times longer than a field multiplication over \( F_p \); 5) computing \( M^d \mod N \) requires an average of 240 1024 bit modular multiplications.

In ITTC scheme, when the ciphertext \( M \) is 1024 bit, the plaintext form of \( M \) can be 1 024 bit. Therefore, we should suppose that the plaintext \( W \) encrypted by a Web client should be 1 024 bit. According to section 2.1, \( W \) should be encoded into 4 integer pairs (the actual length is \( 160 \times 2 \times 4 = 2048 \) bit).

When the Web client encrypts \( W \) and sends the ciphertext to the Web server, the latter will receive 4640 bit ciphertexts, such as \((y, s_1, s_2)\) (1 ≤ \( j \) ≤ 4).

In ITTC scheme, to decrypt \( M \) to some \( t \) share servers and the \( t \) servers will send back different 1 024 bit computation results \( M^d \mod N \). So the total communication cost is about 1024(\( t+1 \)) bits. While in our scheme, when the system decrypts 4640 bit ciphertexts \((y, s_1, s_2)\) (1 ≤ \( j \) ≤ 4), it will broadcast 320 bit \( y \) to some \( t \) share servers and the \( t \) servers will send back different 320 bit computation computed point \( d_y \). So the communication cost is about 320(\( t+1 \)) bits. Therefore, the communication bandwidth our scheme costs is about \( \frac{320(t+1)}{1024(t+1)} \approx 31.2\% \) of that the ITTC scheme costs.

Additionally, in ITTC scheme, the system needs to perform \( t \) times of \( M^d \mod N \) operation and \((t-1)\) modular multiplications. Therefore, the total computation cost of ITTC scheme is

\[
240 \times t + (t-1) = 241t - 1 \text{ times (modular multiplications modulo } N) = \\
(241t - 1) \times 41 = \\
9881t - 41 \text{ times (field multiplications over } F_p) \\
\]

While in our scheme, to decrypt 4640 bit ciphertexts \((y, s_1, s_2)\) (1 ≤ \( j \) ≤ 4), the system needs to perform \( t \) times of \( d_y \) (scalar multiplication) operations, \((t-1)\) point additions, 2 field inversion operations and 8 field multiplications over \( F_p \). So the computation cost of our scheme is

\[
1 \cdot 200 \times t + (t-1) \times (3+2) + 8 \times 3 + 2 = 1 \cdot 205t + 9 \text{ times (field multiplications over } F_p) \\
\]

Because \( t \geq 1 \) definitely, 9881t - 41 > 205t + 9 holds. The computation cost of the ITTC is about 9881t - 41 \( \approx 9.64 \) times greater than that of our scheme.

Therefore, from the efficiency, our scheme also has an advantage to ITTC scheme.

### 5.3 Availability

Our system provides high availability of private keys. Based on \((t, n)\) sharing, even if \((n-t)\) share keys crash (by accident or as a result of an attack) and all data stored them is lost, our system can provide normal service.

When many different clients use the same share servers, the load on the servers may hurt overall system performance. Fortunately, we can choose a random coalition provides for automatic load balancing among the servers. The detailed analysis can be seen in Ref.[2].

As for availability, our scheme is equivalent to the ITTC scheme.

### 5 Related work

Malkin, Wu and Boneh build an intrusion tolerance system by using threshold RSA. Fray[2], Deswarte and Powel and Deswarte, Blain and Fabre describe an encrypted file system where file keys are distributed using Shamir secret sharing across several key servers[8,9]. Keys are reconstructed every time a file is accessed. Jing Jiwu, Feng Dengguo design an Intrusion Tolerant CA Scheme[10]. Kazuo Takaragi, Kunihiko Miyazaki, Masashi Takahashi, et al. study the ECC based threshold Nyberg-Rueppel digital signature scheme[11] and Zhang Xianfeng, Qin Zhiguang, Liu Jinde study the ECC based ElGamal threshold signature scheme[12]. However, there are no any research results on ECC based threshold decryption scheme and intrusion tolerant application reported currently.
7 Conclusions

Rather than prevent or detect intrusions after the fact, we provide some threshold ECC-based intrusion tolerance techniques to build secure Web service. Our scheme can ensure that an attacker who penetrates a few system components cannot compromise total system security.

We haven’t described implementation detail here. We just study some key points to build Web security. In the near future, we will develop a prototype system based on threshold ECC.

References


Brief Introduction to Author(s)

ZHANG Xian-feng (张险峰) was born in 1972. He is now a doctor student. His research interests include information and network security. E-mail: zhangxf@uestc.edu.cn

ZHANG Feng (张 峰) was born in 1977. He is now a doctor student. His research interests include network security, data mining. E-mail: ibmcenter@uestc.edu.cn

QIN Zhi-guang (秦志光) was born in 1956. He is a doctor, professor and supervisor of Ph.D students of UESTC. His research interests include information and network security, intelligent transport system, office automation. E-mail: qinzg@uestc.edu.cn

LIU Jin-de (刘锦德) was born in 1930. He is a professor and supervisor of Ph.D students of UESTC. His research interests include open system and its security, middleware. E-mail: jdliu@uestc.edu.cn