ICIC Express Letters
Volume **10**, Number **9**, September **2016**

 ISSN 1881-803X
pp. **2227–2233**

EMBEDDING MUTUAL PLATFORM-AUTHENTICATION IN THE ENHANCED TRANSPORT LAYER SECURITY PROTOCOL

Yuelei Xiao

Institute of IOT and IT-Based Industrialization
Xi'an University of Posts & Telecommunications
No. 563, South Chang'an Road, Xi'an 710121, P. R. China
xiao_yuelei@163.com

Received February 2016; accepted May 2016

Abstract. *Based on the Enhanced Transport Layer Security (ETLS) and existing mutual Platform-Authentication protocols, different Trusted Network Connect (TNC) framework solutions can be formed to realize mutual user authentication and Platform-Authentication. However, they need more communication costs. To solve this problem, the ETLS protocol is further enhanced by embedding mutual Platform-Authentication in it. The enhanced ETLS is backward compatibility with the TLS and ETLS protocols, and secure in the extended Strand Space Model (SSM).*
Keywords: Transport layer security, Platform-authenticaion, Strand space model

1. **Introduction.** Transport Layer Security (TLS) protocol [1] is to provide privacy and data integrity between two communicating entities. The TLS protocol is composed of a TLS record protocol and a TLS handshake protocol. With the rapid development of network technologies, new security challenges have been introduced. One of these security challenges concerns the increasing need for Platform-Authentication [2]. And some mutual Platform-Authentication protocols based on the ISO/IEC 9798-3:1998/Amd 1:2010 were proposed [3-6]. To support these mutual Platform-Authentication protocols, an Enhanced TLS (ETLS) protocol was given in [7]. The ETLS protocol can not only realize mutual user authentication, but also establish three secure channels to support those mutual Platform-Authentication protocols. However, it does not include the function of mutual Platform-Authentication, so it needs to be combined with those mutual Platform-Authentication protocols to form different Trusted Network Connect (TNC) framework [2] solutions. These solutions can simultaneously realize mutual user authentication and Platform-Authentication, while they need more communication costs [2].

To solve this problem, the ETLS protocol is further enhanced by embedding mutual Platform-Authentication in it. Consequently, the enhanced ETLS protocol can simultaneously realize mutual user authentication and Platform-Authentication between the client and server. And it can also establish three secure channels to support mutual Platform-Authentication, including the secure channel between the client and server, the secure channel between the client and online Trusted Center (TC), and the secure channel between the server and online TC. Moreover, it is backward compatibility with the TLS and ETLS protocols, and secure in the extended Strand Space Model (SSM) [8]. The remainder of this letter is organized as follows. Section 2 proposes an enhanced ETLS protocol. Section 3 discusses its backward compatibility and security. And finally Section 4 provides the conclusions.

2. **The Enhanced ETLS Protocol.** The enhanced ETLS protocol concerns a client, a server and an online TC. And it includes a record protocol and a handshake protocol,

like the TLS and ETLS protocols. Its record protocol provides confidentiality and data integrity between the client and the server, similar to the TLS and ETLS record protocols except its version field. And its handshake protocol can be used to establish the secure channel between the client and server, the secure channel between the client and online TC, and the secure channel between the server and online TC. Moreover, its handshake protocol can be used to realize mutual user authentication and Platform-Authentication between the client and server. In its handshake protocol, the handshake messages between the client and server are encapsulated in its record protocol, while the handshake messages between the server and online TC are encapsulated in some reliable transport protocols (e.g., UDP). The TLS, ETLS and enhanced ETLS handshake protocols with optional messages marked are shown in Figure 1.

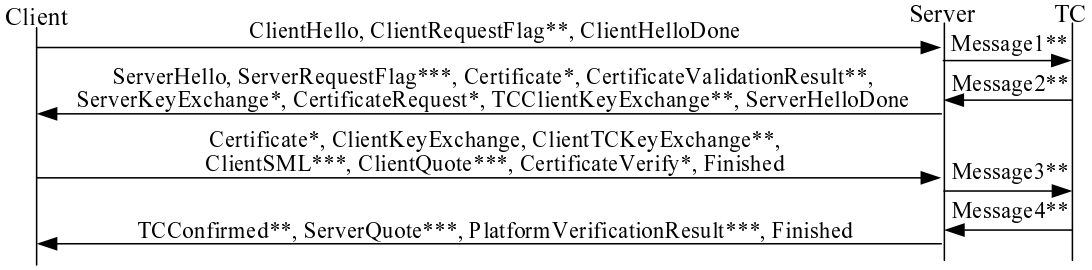

FIGURE 1. The TLS, ETLS and enhanced ETLS handshake protocols with optional messages marked

In Figure 1, all the optional messages are marked with one star, two stars or three stars. For the TLS handshake protocol, the optional messages are marked with one star. And for the ETLS handshake protocol, nine optional messages are added and marked with two stars. In the enhanced ETLS handshake protocol, another five optional messages are added and marked with three stars, and some existing optional messages need to be extended. The messages of the enhanced ETLS handshake protocol are as follows.

(1) m_1: ClientHello, including N_C, $Ciphers_C$, etc., where N_C is a random number generated by the client and $Ciphers_C$ is a list of cipher suites supported by the client. If the client wants to validate the server's certificate using the online TC, then m_1 does not include a CA list trusted by the client.

(2) m_2: ClientRequestFlag. If the client wants to validate the server's certificate using the online TC or establish a secure channel with the online TC, or perform Platform-Authentication of the server, it sends m_2 to the server, which includes an 8-bit string $flag_1$. If the first bit of $flag_1$ is 1, then the client needs to validate the server's certificate using an online TC; otherwise it does not need. If the second bit of $flag_1$ is 1, then the client needs to establish a secure channel with the online TC; otherwise it does not need. If the third bit of $flag_1$ is 1, then the client needs to perform Platform-Authentication of the server; otherwise it does not need.

(3) m_3: ClientHelloDone, indicating the end of the ClientHello and associated messages.

(4) M_1: Message1. If the client wants to validate the server's certificate using the online TC or establish a secure channel with the online TC, or the server wants to establish a secure channel with the online TC, the server sends M_1 to the online TC. M_1 includes $flag_2$, N_S and $Ciphers_S$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 0 and 1 respectively. M_1 includes $flag_2$, N_C and $Ciphers_C$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 1 and 0 respectively. M_1 includes $flag_2$, N_C, $Ciphers_C$, N_S and $Ciphers_S$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 1 and 1 respectively. M_1 includes $flag_2$, N_C and $Certs_S$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 0 and 0 respectively. M_1 includes $flag_2$, N_C, $Certs_S$, N_S and $Ciphers_S$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 0 and 1 respectively. M_1

includes $flag_2$, N_C, $Cert_S$ and $Ciphers_C$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 1 and 0 respectively. M_1 includes $flag_2$, N_C, $Cert_S$, $Ciphers_C$, N_S and $Ciphers_S$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 1 and 1 respectively. In M_1, $flag_2$ is an 8-bit string and $Ciphers_S$ is a list of cipher suites supported by the server. If the first bit of $flag_2$ is 1, then the client needs to validate the server's certificate using an online TC; otherwise it does not need. If the third bit of $flag_2$ is 1, then the client needs to establish a secure channel with the online TC; otherwise it does not need. If the fourth bit of $flag_2$ is 1, then the server needs to establish a secure channel with the online TC; otherwise it does not need. Note that if the third bit of $flag_1$ is 1, then the fourth bit of $flag_2$ is set to 1.

(5) M_2: Message2. On receipt of M_1, the online TC sends M_2 to the server. M_2 includes $flag_2$, N_T, $Ciphers_{T,S}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 0 and 1 respectively. M_2 includes $flag_2$, N_T, $Ciphers_{T,C}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 1 and 0 respectively. M_2 includes $flag_2$, N_T, $Ciphers_{T,C}$, $Ciphers_{T,S}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 0, 1 and 1 respectively. M_2 includes $flag_2$, Re_S and $\sigma_{T,1}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 0 and 0 respectively. M_2 includes $flag_2$, Re_S, $\sigma_{T,1}$, N_T, $Ciphers_{T,S}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 0 and 1 respectively. M_2 includes $flag_2$, Re_S, $\sigma_{T,1}$, N_T, $Ciphers_{T,C}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 1 and 0 respectively. M_2 includes $flag_2$, Re_S, $\sigma_{T,1}$, N_T, $Ciphers_{T,C}$, $Ciphers_{T,S}$, g^z and $\sigma_{T,2}$ if the first bit, the third bit and the fourth bit of $flag_2$ are 1, 1 and 1 respectively. In M_2, Re_S indicates whether or not $Cert_S$ is valid, $\sigma_{T,1}$ is a signature of the online TC and $\sigma_{T,1} = [N_C||Cert_S||Re_S]_{sk_T}$, N_T is a random number generated by the online TC, $Ciphers_{T,C}$ is a cipher suite selected by the online TC for the client and online TC, $Ciphers_{T,S}$ is a cipher suite selected by the online TC for the server and online TC, g^z is a temporal public key of the online TC, $\sigma_{T,2}$ is a signature of the online TC and $\sigma_{T,2} = [g^z]_{sk_T}$, and sk_T is a signing key of the online TC.

(6) m_4: ServerHello, including N_S, $Ciphers_{S,C}$, etc., where N_S is a random number generated by the server and $Ciphers_{S,C}$ is a cipher suite selected by the server.

(7) m_5: ServerRequestFlag. If the server wants to perform Platform-Authentication of the client, it sends m_5 to the client, which includes an 8-bit string $flag_3$. If the first bit of $flag_3$ is 1, then the server needs to perform Platform-Authentication of the client; otherwise it does not need. Note that the first bit of $flag_3$ can be set to 1 when the second bit of $flag_1$ is 1.

(8) m_6: Certificate sent by the server, only including a certificate of the server $Cert_S$ if the client wants to validate the server's certificate using the online TC.

(9) m_7: CertificateValidationResult. If the first bit of $flag_1$ is 1, then the server sends m_6 to the client, which includes Re_S and $\sigma_{T,1}$.

(10) m_8: ServerKeyExchange, including g^y and $\sigma_{S,1}$, where g^y is a temporal public key of the server, $\sigma_{S,1}$ is a signature of the server and $\sigma_{S,1} = [g^y]_{sk_S}$, and sk_S is a signing key of the server.

(11) m_9: CertificateRequest, which does not include a CA list trusted by the server if the server wants to validate the client's certificate using the online TC.

(12) m_{10}: TCClientKeyExchange. If the second bit of $flag_1$ is 1, then the server sends m_9 to the client, which includes N_T, $Ciphers_{T,C}$, g^z and $\sigma_{T,2}$.

(13) m_{11}: ServerHelloDone, indicating the end of the ServerHello and associated messages.

(14) m_{12}: Certificate sent by the client, only including a certificate of the client $Cert_C$ if the server wants to validate the client's certificate using the online TC.

(15) m_{13}: ClientKeyExchange, including a temporal public key of the client g^x.

(16) m_{14}: ClientTCKeyExchange. If the second bit of $flag_1$ is 1, then the client sends m_{13} to the server. m_{13} includes $\sigma_{C,1}$ and $MAC_{C,1}$, where $\sigma_{C,1}$ is a signature of the client and $\sigma_{C,1} = [g^x]_{sk_C}$, $MAC_{C,1}$ is a message authentication code of the client and $MAC_{C,1} = hash(K_1, N_C||Ciphers_C||N_T||Ciphers_{T,C}||g^z||\sigma_{T,2}||Cert_C||g^x||\sigma_{C,1})$, sk_C is a signing key of the client, $K_1 = hash(g^{xz}, N_C||N_T)$ and is a session key between the client and online TC.

(17) m_{15}: ClientSML. If the first bit of $flag_3$ is 1, then the client sends m_{15} to the server. m_{15} includes $\{SML_\alpha\}_{K_1}$, where SML_α is some SMLs of α, and α is the platform of the client.

(18) m_{16}: ClientQuote. If the first bit of $flag_3$ is 1, then the client sends m_{16} to the server. m_{16} includes PCR_α, $Cert(AIK_{pub,\alpha})$ and σ_α, where PCR_α is some Platform Configuration Registers (PCRs) of α, $Cert(AIK_{pub,\alpha})$ is an AIK certificate of α, σ_α is an AIK signature of α and $\sigma_\alpha = [hash(MK, N_S)||PCR_\alpha]_{AIK_{priv,\alpha}}$, and $AIK_{pub,\alpha}$ and $AIK_{priv,\alpha}$ are an AIK public and private key pair.

(19) m_{17}: CertificateVerify, including a signature of the client $\sigma_{C,2}$ and $\sigma_{C,2} = [hms_1]_{sk_C}$, where hms_1 is the handshake messages from m_1 to m_{16}.

(20) m_{18}: Finished sent by the client, including a message authentication code of the client $MAC_{C,2}$ and $MAC_{C,2} = hash(MK, hms_2)$, where $MK = hash(g^{xy}, N_C||N_S)$ and is a master key between the client and server, and hms_2 is the handshake messages from m_1 to m_{17}.

(21) M_3: Message3. If the server wants to validate the client's certificate using the online TC or establish a secure channel with the online TC or perform Platform-Authentication of the client, or the client wants to establish a secure channel with the online TC or perform Platform-Authentication of the server, the server sends M_3 to the online TC. M_3 includes $flag_2$, $Cert_S$, g^y, $\sigma_{S,1}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 0, 1, 0 and 0 respectively. M_3 includes $flag_2$, $Cert_C$, g^x, $\sigma_{C,1}$ and $MAC_{C,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 0, 0 and 0 respectively. M_3 includes $flag_2$, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, $Cert_S$, g^y, $\sigma_{S,1}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 1, 0 and 0 respectively. M_3 includes $flag_2$, N_S and $Cert_C$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 0, 0 and 0 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, $Cert_S$, g^y, $\sigma_{S,1}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 1, 0 and 0 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, g^x, $\sigma_{C,1}$ and $MAC_{C,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 0, 0 and 0 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, $Cert_S$, g^y, $\sigma_{S,1}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 1, 0 and 0 respectively. M_3 includes $flag_2$, $Cert_S$, g^y, $\sigma_{S,1}$, N_C, PCR_β, $Cert(AIK_{pub,\beta})$, $\{SML_\beta\}_{K_2}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 0, 1, 0 and 1 respectively. M_3 includes $flag_2$, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, N_S, PCR_α, $Cert(AIK_{pub,\alpha})$, and $\{SML_\alpha\}_{K_1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 0, 1 and 0 respectively. M_3 includes $flag_2$, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, $Cert_S$, g^y, $\sigma_{S,1}$, PCR_β, $Cert(AIK_{pub,\beta})$, $\{SML_\beta\}_{K_2}$, PCR_α, $Cert(AIK_{pub,\alpha})$, $\{SML_\alpha\}_{K_1}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 1, 1 and 1 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, $Cert_S$, g^y, $\sigma_{S,1}$, N_C, PCR_β, $Cert(AIK_{pub,\beta})$, $\{SML_\beta\}_{K_2}$ and $MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 1, 0 and 1 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, PCR_α, $Cert(AIK_{pub,\alpha})$, and $\{SML_\alpha\}_{K_1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 0, 1 and 0 respectively. M_3 includes $flag_2$, N_S, $Cert_C$, g^x, $\sigma_{C,1}$, $MAC_{C,1}$, $Cert_S$, g^y, $\sigma_{S,1}$, PCR_β, $Cert(AIK_{pub,\beta})$, $\{SML_\beta\}_{K_2}$, PCR_α, $Cert(AIK_{pub,\alpha})$, $\{SML_\alpha\}_{K_1}$ and

$MAC_{S,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 1, 1 and 1 respectively. In M_3, SML_β is some SMLs of β, β is the platform of the server, PCR_β is some PCRs of β, $Cert(AIK_{pub,\beta})$ is an AIK certificate of β, $AIK_{pub,\beta}$ is an AIK public key, $MAC_{S,1}$ is a message authentication code of the server and $MAC_{S,1} = hash(K_2,\ M_1||M_2||M_3{}^o)$, $K_2 = hash(g^{yz},\ N_S||N_T)$ and is a session key between the server and online TC, and $M_3{}^o$ is M_3 except $MAC_{S,1}$. If the second bit of $flag_2$ is 1, then the server needs to validate the client's certificate using an online TC; otherwise it does not need. If the fifth bit of $flag_2$ is 1, then the server needs to perform Platform-Authentication of the client; otherwise it does not need. If the sixth bit of $flag_2$ is 1, then the client needs to perform Platform-Authentication of the server; otherwise it does not need. Note that if the third bit of $flag_1$ is 1, then the sixth bit of $flag_2$ is set to 1. And if the first bit of $flag_3$ is 1, then the fifth bit of $flag_2$ is set to 1.

(22) M_4: Message4. On receipt of M_3, the online TC sends M_4 to the server. M_4 includes $flag_2$ and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 0, 1, 0 and 0 respectively. M_4 includes $flag_2$ and $MAC_{T,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 0, 0 and 0 respectively. M_4 includes $flag_2$, $MAC_{T,1}$ and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 1, 0 and 0 respectively. M_4 includes $flag_2$, Re_C and $\sigma_{T,3}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 0, 0 and 0 respectively. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$ and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 1, 0 and 0 respectively. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$ and $MAC_{T,1}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 0, 0 and 0 respectively. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$, $MAC_{T,1}$ and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 1, 0 and 0 respectively. M_4 includes $flag_2$, Res_T, σ_T and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 0, 1, 0 and 1 respectively, where $Res_T = N_C||PCR_\beta||Cert(AIK_{pub,\beta})||Re_{AIK,\beta}||Re_{INT,\beta}$. M_4 includes $flag_2$, $MAC_{T,1}$, Res_T, and σ_T if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 0, 1 and 0 respectively, where $Res_T = N_S||PCR_\alpha||Cert(AIK_{pub,\alpha})||Re_{AIK,\alpha}||Re_{INT,\alpha}$. M_4 includes $flag_2$, $MAC_{T,1}$, Res_T, σ_T and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 0, 1, 1, 1 and 1 respectively, where $Res_T = N_S||\,PCR_\alpha||\,Cert(AIK_{pub,\alpha})||\,Re_{AIK,\alpha}||\,Re_{INT,\alpha}||\,N_C||\,PCR_\beta||\,Cert(AIK_{pub,\beta})||\,Re_{AIK,\beta}||\,Re_{INT,\beta}$. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$, Res_T, σ_T and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 0, 1, 0 and 1 respectively, where $Res_T = N_C||PCR_\beta||Cert(AIK_{pub,\beta})||Re_{AIK,\beta}||Re_{INT,\beta}$. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$, $MAC_{T,1}$, Res_T, and σ_T if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 0, 1 and 0 respectively, where $Res_T = N_S||PCR_\alpha||Cert(AIK_{pub,\alpha})\ ||Re_{AIK,\alpha}||Re_{INT,\alpha}$. M_4 includes $flag_2$, Re_C, $\sigma_{T,3}$, $MAC_{T,1}$, Res_T, σ_T and $MAC_{T,2}$ if the second bit, the third bit, the fourth bit, the fifth bit and the sixth bit of $flag_2$ are 1, 1, 1, 1 and 1 respectively, where $Res_T = N_S||\,PCR_\alpha\,||\,Cert(AIK_{pub,\alpha})||\,Re_{AIK,\alpha}||\,Re_{INT,\alpha}||\,N_C||\,PCR_\beta||\,Cert(AIK_{pub,\beta})||Re_{AIK,\beta}||\,Re_{INT,\beta}$. In M_4, σ_T is a signature of the online TC and $\sigma_T = [Res_T]_{sk_T}$, $Re_{AIK,\alpha}$ and $Re_{AIK,\beta}$ are AIK certificate validation results of $Cert(AIK_{pub,\alpha})$ and $Cert(AIK_{pub,\beta})$ respectively, $Re_{INT,\alpha}$ and $Re_{INT,\beta}$ are platform integrity verification results of SML_α and SML_β respectively, Re_C indicates whether or not $Cert_C$ is valid, $\sigma_{T,3}$ is a signature of the online TC and $\sigma_{T,3}=[N_S||Cert_C||Re_C]_{sk_T}$, $MAC_{T,1}$ is a message authentication code of the online TC and $MAC_{T,1} = hash(K_1, N_C||Ciphers_C||N_T||Ciphers_{T,C}||g^z||\sigma_{T,2}||Cert_C||g^x||\sigma_{C,1}||MAC_{C,1})$, $MAC_{T,2} = hash(K_2, M_1||M_2||M_3||M_4{}^o)$, and $M_4{}^o$ is M_4 except $MAC_{T,2}$.

(23) m_{19}: TCConfirmed. If the second bit of $flag_1$ is 1, then the server sends m_{16} to the client, which includes $MAC_{T,1}$.

(24) m_{20}: ServerQuote. If the third bit of $flag_1$ is 1, then the server sends m_{20} to the client. m_{20} includes PCR_β, $Cert(AIK_{pub,\beta})$ and σ_β, where σ_β is an AIK signature of β and $\sigma_\beta = [hash(MK, N_C)||PCR_\beta]_{AIK_{priv,\beta}}$, and $AIK_{priv,\beta}$ is an AIK private key.

(25) m_{21}: PlatformVerificationResult. If the third bit of $flag_1$ is 1, then the server sends m_{21} to the client. m_{21} includes Res_T and σ_T.

(26) m_{22}: Finished sent by the server, including a message authentication code of the server $MAC_{S,2}$ and $MAC_{S,2} = hash(MK, hms_3)$, where hms_3 is the handshake messages from m_1 to m_{21}.

3. **Discussions.** In [7], the ETLS protocol is proved to be backward compatibility with the TLS protocol. Compared with the ETLS protocol, the enhanced ETLS protocol adds another five optional messages, which are shown in Figure 1. And it extends some optional messages by adding another control bits and data fields in them. These extended messages include the ClientRequestFlag, Message3 and Message4 messages. Hence, it is backward compatibility with the TLS and ETLS protocols. On the whole, the enhanced ETLS protocol only adds another five messages and extends another three messages to realize mutual Platform-Authentication, so it is also effective.

According to [7], the ETLS handshake protocol is proved secure in the SSM [9]. And it can implement mutual user authentication and simultaneously establish three secure channels, including the secure channel between the client and server, the secure channel between the client and online TC, and the secure channel between the server and online TC. In the enhanced ETLS protocol, the same functions as the ETLS protocol are also implemented. And the secure channel between the client and server is cryptographically bound with the client's and server's AIK signatures respectively, similar to the improved MN-TAP protocol described in [8]. Because the improved MN-TAP protocol was proved secure in the extended SSM [8], the enhanced ETLS protocol is also secure in the extended SSM. Moreover, the secure channel between the client and online TC is used to encrypt the client's Store Measurement Logs (SMLs), and the secure channel between the server and online TC is used to encrypt the server's SMLs, so the enhanced ETLS protocol can prevent the client's and server's platform configuration privacy being leaked between the client and server. Hence, the enhanced ETLS protocol is secure.

4. **Conclusions.** In this letter, the ETLS protocol is further enhanced by embedding mutual Platform-Authentication in it. The enhanced ETLS protocol can realize mutual user authentication and establish three secure channels, similar to the ETLS protocol. And it can also realize mutual Platform-Authentication. That is to say, it can simultaneously realize mutual user authentication and Platform-Authentication. Moreover, it is backward compatibility with the TLS and ETLS protocols, and secure in the extended SSM. In the future, we will further enhance traditional security protocols by embedding mutual Platform-Authentication in them.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (61402367). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, 2008.

[2] TCG, *TCG TNC Architecture for Interoperability Specification Version 1.4*, TCG, 2009.

[3] Y. Xiao, Y. Wang and L. Pang, A mutual integrity reporting scheme for remote attestation, *China Communications*, vol.5, pp.165-165, 2010.

[4] Y. Xiao and Y. Wang, WLAN access authentication schemes in trusted environment, *Journal of Lanzhou University (Natural Sciences)*, vol.49, no.4, pp.547-553, 2013.

[5] Y. Xiao, Y. Wang, L. Pang and S. Tan, WLAN Mesh security association scheme in trusted computing environment, *Journal on Communications*, vol.35, no.7, pp.94-103, 2014.
[6] ISO/IEC, *ISO/IEC 9798-3:1998/Amd.1:2010 Information Technology – Security Techniques – Entity Authentication – Part 3: Mechanisms Using Digital Signature Techniques AMENDMENT 1*, 2010.
[7] Y. Xiao, Z. Zhu and Y. Zhang, An enhanced transport layer security protocol to support mutual platform-authentication protocols, *ICIC Express Letters*, vol.9, no.9, pp.2369-2374, 2015.
[8] Y. Xiao, Y. Wang and L. Pang, Verification of trusted network access protocols in the strand space model, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol.E95-A, no.3, pp.665-668, 2012.
[9] J. C. Herzog, The Diffie-Hellman key-agreement scheme in the strand-space model, *Proc. of the 16th IEEE Computer Security Foundation Workshop*, Pacific Grove, USA, pp.234-247, 2003.