Key Transport Protocol Based On Hybryd Encryption Scheme

Alex Yu. Nesterenko*

Abstract

We present a new variant of key transport protocol based on hybryd encryption scheme. We combine this protocol with simple transport protocol and obtain a realization of secure socket layer. The security of this combination is based on the security of symmetric encryption and discrete logarithm problem.

**Keywords**: hybryd encryption, symmetric encryption, digital signature, message authentification and elliptic curves.

Suppose two entities, say \( A \) and \( B \), want to organize secure communication over public network. We assume that all entities have public domain parameters — elliptic curve \( E \) defined over the field \( \mathbb{F}_p \) and some point \( P \in E \) which generates a subgroup of prime order \( q \).

Let entity \( A \) has an asymmetric signature key pair \((x_A, Y_A)\). These keys and signature/verification process are described by russian goverment standard GOST R 34.10.

Let entity \( B \) has an assymetric key pair \((x_B, Y_B)\), where the secret key \( x_B \in \mathbb{Z}_q \), and the public key \( Y_B \) is the point of the elliptic curve \( E \) satisfying an equation \( Y_B = [x_B]P \) for the point \( P \in E \).

Entity \( A \) generates the common key \( CK \) and encrypts it with entity \( B \)'s public encryption transformation as described in [1], i.e. entity \( A \)

- generates two random integers \( k, \xi_0 \), where \( 0 < k < q \), \( \xi_0 \in \mathbb{Z}_m \) and \( m \) is a bit length of session key \( CK \),
- generates two points \( U = [k]P \), \( W = [k]Y_B \),
- generates a session key \( K = Kdf(W, \xi_0) \),

and sends a message \( M_0 \) with encrypted common key \( CK \) and some text \( T_0 \) to entity \( B \)

\[
M_0 = U||Enc(K, CK||\xi_0||T_1)||Sign(x_A, U||CK||\xi_0||T_0)\
\]

*National Research University “Higher Scholl Of Economics”, Moscow, Russia.
Entity $B$ gets the message $M_0$ and checks the signature. If signature is true, entity $B$ calculates the point $W = [x_B]U$ and the session key $K = Kdf(W, \xi_0)$. After that it decrypts the common key $CK$ and $T_0$.

Since entity $A$ wants to make sure that entity $B$ has a common key $CK$, entity $B$ calculates a random value $\xi_1 \in \mathbb{Z}_m$ and session identifier $SID = Kdf(\xi_0, \xi_1)$. After that it sends a message $M_1$ and some text $T_1$ to entity $A$

$$M_1 = H_1||Enc(CK, \xi_1||T_1)||Mac(CK, H_1||\xi_1||T_1),$$

where $H_1 = SID||Hash(\xi_0)$.

Presented protocol is most similar to the second secret key transport mechanism from [2, sect. 12.2]. But we have some important differences.

- Our protocol has two passes.
- All authentication codes are calculated before the data encryption.
- We calculate authentication codes for all transmitted information.
- We use a sequence of random numbers $\xi_0, \xi_1, \ldots$ unlike synchronous clocks or time stamps. Moreover, all values of this sequence are transmitted in the encrypted form.

The presence of two passes is explained by the further exchange of encrypted data. A simple transport protocol consists of the following steps. Let $Data_{2n}$ is the $n$-th data octet which should be send from entity $A$ to entity $B$. Entity $A$ generates a random integer $\xi_{2n} \in \mathbb{Z}_m$ and sends the message $M_{2n}$ to entity $A$

$$M_{2n} = H_{2n}||Enc(CK, \xi_{2n}||Data_{2n})||Sign(x_A, H_{2n}||\xi_{2n}||Data_{2n}),$$

where $H_{2n} = SID||Hash(\xi_{2n-1})$ for all $n = 1, 2, \ldots$.

Entity $B$ gets this message and checks the signature. In case it false the connection is dropped, entity $B$ destroys common key $CK$ and session identifier $SID$.

If the signature is true, entity $B$ calculates a random integer $\xi_{2n+1} \in \mathbb{Z}_m$ and return some information $Data_{2n+1}$ to entity $A$. It sends a message

$$M_{2n+1} = H_{2n+1}||Enc(CK, \xi_{2n+1}||Data_{2n+1})||Mac(CK, H_{2n+1}||\xi_{2n+1}||Data_{2n+1}),$$

where $H_{2n+1} = SID||Hash(\xi_{2n})$.

References
