

AN INVERSIONLESS DL/EC SIGNATURE FACILITATING CHAINING

Benjamin Arazi
CipherIT
38 Sigalon Str.
Omer. ISRAEL
arazi@ee.bgu.ac.il

1. An inversionless DL/EC signature

Let x denote the signer's private key. Let \mathbf{G} be a generating point. Let \mathbf{Y} denote the signer's public key, where $\mathbf{Y} = x*\mathbf{G}$ (we use EC notations, where $*$ denotes an EC exponentiation). Let $H(v, \mathbf{W})$ be a hash transformation that converts a scalar v and a point \mathbf{W} into a scalar.

Signature generation

To sign a message m , the signer generates a random k , calculates $\mathbf{T} = k*\mathbf{G}$ and $s = H(m, \mathbf{T})*k + x$

(All scalar calculations are performed modulo the order of \mathbf{G} .)

The signature is the pair $\{\mathbf{T} ; s\}$.

The signer submits $m, \mathbf{T}, s, \mathbf{Y}$.

Signature verification

The verifier checks that $s*\mathbf{G} = H(m, \mathbf{T})*\mathbf{T} + \mathbf{Y}$.

It is noted that neither the signer nor the verifier performs an inverse calculation modulo the order of \mathbf{G} .

2. Chained procedures

2.1. Hierarchical key-issuing.

We treat a procedure in which a $User_i$ issues a private key $x_{(i+1)}$ and a public value $U_{(i+1)}$ to $User_{(i+1)}$, whose identification details are $ID_{(i+1)}$, where the CA acts as $User_0$.

While $x_0 = \log U_0$, x_i is not $\log U_i$ for $i > 0$, and $User_i$ does not know $\log U_i$.

For ease of explanation, we treat a procedure in which a key-issuer knows the private key which he issues to the next user in line. The process can be extended to the case where an issued private key is not known to the issuer.

$User_i$ generates a random $h_{(i+1)}$ and issues to $User_{(i+1)}$ the public value $U_{(i+1)} = h_{(i+1)} * G$ and the private key $x_{(i+1)} = H(ID_{(i+1)}, U_{(i+1)}) * h_{(i+1)} + x_i$.

In order to establish the validity of the values $U_{(i+1)}$ and $x_{(i+1)}$ issued to him, $User_{(i+1)}$ must be provided, by definition, with reference information regarding all the users in the chain which preceded him, up to the CA. In the current case, this reference information consists of ID_j and U_j , $j = 0, 1, 2, \dots, i$ (where ID_0 identifies the CA). It is noted that there is no explicit certificate which attests to the association between ID_j and U_j .

Based on this reference information, $User_{(i+1)}$ checks whether $x_{(i+1)} * G = H(ID_{(i+1)}, U_{(i+1)}) * U_{(i+1)} + H(ID_i, U_i) * U_i + H(ID_{(i-1)}, U_{(i-1)}) * U_{(i-1)} + \dots + H(ID_1, U_1) * U_1 + U_0$.

The described chained certificate verification requires a total of $i+2$ exponentiations, where the public values are self-certified. In comparison, in standard certificate verification a verifier would perform $2(i+1)$ exponentiations (associated with $i+1$ separate DL/EC signature verifications), where each verification refers to an explicit certificate.

*The presented chaining is facilitated by the fact that the private key x_i of $User_i$ is used by that user as a free addend in the generation of the private key $x_{(i+1)}$. (That is, x_i is not multiplied by any scalar, as seen in the expression $x_{(i+1)} = H(ID_{(i+1)}, U_{(i+1)}) * h_{(i+1)} + x_i$.)*

2.2. Hierarchical signature generation and verification.

User_i signs a message m according to the method presented before. That is, User_i generates $\mathbf{T} = k * \mathbf{G}$ for a random k , and $s = H(m, \mathbf{T}) * k + x_i$. The signature is the pair $\{s, \mathbf{T}\}$.

Using ID_j and U_j , $j = 0, 1, 2, \dots, i$, a verifier checks that

$$s * \mathbf{G} = H(m, \mathbf{T}) * \mathbf{T} + H(ID_i, U_i) * U_i + H(ID_{(i-1)}, U_{(i-1)}) * U_{(i-1)} + \dots + H(ID_1, U_1) * U_1 + U_0.$$

Here again, the chained verification requires a total of $i+2$ exponentiations, where the public values are self-certified, compared to a standard verification which would require $2(i+1)$ exponentiations associated with $i+1$ separate DL/EC signature verifications, with an explicit certificate associated with each user.