# Elliptic Curve Digital Signatures and Accessories

Lein Harn*, Guang Gong**

*Department of Computer Networking
University of Missouri – Kansas City
Kansas City, MO 64110
TEL:(816)235-2358
FAX:((816)235-5159
Email:HARN@CSTP.UMKC.EDU
USA

** Department of Combinatorics & Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

**<Abstract>**

Digital signatures have been used in Internet applications to provide data authentication and non-repudiation services. Digital signatures will keep on playing an important role in future Internet applications. There are two most well-known public-key cryptosystems, the RSA scheme and the ElGamal scheme, which can provide both digital signature and data encryption. More recently, the Elliptic Curve Cryptosystem (ECC), in which the difficulty of breaking the system is based on the difficulty of computing a discrete logarithm over an elliptic curve, has also been considered to become a standard in the IEEE P1363 project. The ECC shares many similar features with the ElGamal cryptographic system. In this paper, we want to show that, (a) there are variants of elliptic curve digital signatures, and (b) in addition to provide non-repudiation services, there are several interesting accessories associated with discrete-logarithm based digital signatures. These accessories have important features, such as providing subliminal channels, providing multisignatures, and providing batch verification, etc. In better understanding of these features, we can select digital signature schemes with additional features.

## 1. Introduction

A digital signature is analogous to an ordinary hand-written signature used for signing messages. It must be unique and private to the signer. More specifically, suppose that B is the recipient of a message *m* signed by A. Then, A's signature must satisfy three requirements [1]:

1. B must be able to validate A's signature on *m* easily;
2. it must be impossible for anyone, including B, to forge A's signature; and
3. it must be possible for a judge or third party to resolve any dispute between A and B.

At this time, there are two most popular public-key algorithms that can provide digital signatures: (a) the RSA scheme [2], in which the difficulty of breaking the scheme is based on solving the factoring a large integer into two large prime factors, and (b) the ElGamal scheme [3], in which the difficulty of breaking the scheme is based on solving the discrete logarithm problem. In 1991, the US government has proposed the Digital Signature Standard (DSS) as a federal standard to enable federal government agencies to use the Digital Signature Algorithm (DSA) [4] to sign electronic documents. The DSA is one of the ElGamal-type signature schemes based on the discrete logarithm problem. More recently, the Elliptic Curve Cryptosystem (ECC), in which the difficulty of breaking the system is based on the difficulty of computing a discrete logarithm over an elliptic curve, has also been considered to become a standard in the IEEE P1363 project. The ECC shares many similar features with the ElGamal cryptographic system.

Since digital signature has one of the unique features associated with the public-key cryptography, digital signature has been used in security services to provide non-repudiation services. For example, digital signature has been used in the Secure Electronic Transactions (SET) standard [5] to provide security of electronic transfers of credit and payment information over the Internet. Digital signature has been adopted by many security protocols, such as SSL [6] and ISAKMP-OAKLEY [7], to provide data authentication and non-repudiation services. Digital signature will keep on playing an important role in future Internet applications.

The purpose of this paper is to show that (a) there are variants of elliptic curve digital signatures, and (b) in addition to provide non-repudiation services, there are several interesting accessories associated with discrete-logarithm based digital signature. In particular, we will focus on the elliptic curve signatures. Each of these accessories has one unique feature. These features can be fully utilized to create additional services of Internet applications. For example, we will show that some digital signatures can support efficient batch verification, some signatures exist covert channels to hide secret messages, and some signatures can be combined efficiently to produce a group signature. Our discussion will be based on the generalized ElGamal-type that was published in 1994 [8]. We will discuss each accessory separately and identify elliptic curve signature variants that can provide such service.

## 2. Generalized ElGamal-type Digital Signatures (18 variants)

Harn and Xu had published a generalized ElGamal-type digital signature schemes in 1994 [8]. In their paper, they proposed a systematic design of digital signature schemes. Instead of signing the message *m* directly, all ElGamal-type signature schemes should sign the one-way hash result of *m*. For simplicity, we will ignore the one-way hash function in the following discussion. In ElGamal original scheme, there are five

parameters {*r, s, m, k, x*} in the signature signing equation, where {*r, s*} is the digital signature of message *m*, *k and x* are secrets of the signer. By properly permuting these five parameters, they have come up 18 secure variants. We list these 18 ElGamal-type variants in Table 1. We would like to point out here that, although additional generalizations and variations can generate more than 13,000 variants [9], but this 18 variants are the most efficient variants since each of these 18 variants permutes these five parameters {*r, s, m, k, x.*} directly. In addition, these 18 variants include all previously found ElGamal-type simple variants.

Since DSA is a special form of the original ElGamal scheme, which utilizes another parameter *q,* where *q* is a prime factor of *p-1*, all 18 ElGamal-type variants can be easily converted to DSA-type variants.

| equation number | signature equation | signature verification |
|---|---|---|
| 1 | $mx=rk+s \mod p-1$ | $y^m=r^r\alpha^s \mod p$ |
| 2 | $mx=sk+r \mod p-1$ | $y^m=r^s\alpha^r \mod p$ |
| 3 | $rx=mk+s \mod p-1$ | $y^r=r^m\alpha^s \mod p$ |
| 4 | $rx=sk+m \mod p-1$ | $y^r=r^s\alpha^m \mod p$ |
| 5 | $sx=rk+m \mod p-1$ | $y^s=r^r\alpha^m \mod p$ |
| 6 | $sx=mk+r \mod p-1$ | $y^s=r^m\alpha^r \mod p$ |
| 7 | $rmx=k+s \mod p-1$ | $y^{rm}=r\alpha^s \mod p$ |
| 8 | $x=mrk+s \mod p-1$ | $y=r^{mr}\alpha^s \mod p$ |
| 9 | $sx=k+mr \mod p-1$ | $y^s=r\alpha^{mr} \mod p$ |
| 10 | $x=sk+rm \mod p-1$ | $y=r^s\alpha^{rm} \mod p$ |
| 11 | $rmx=sk+1 \mod p-1$ | $y^{rm}=r^s\alpha \mod p$ |
| 12 | $sx=rmk+1 \mod p-1$ | $y^s=r^{rm}\alpha \mod p$ |
| 13 | $(r+m)x=k+s \mod p-1$ | $y^{r+m}=r\alpha^s \mod p$ |
| 14 | $x=(m+r)k+s \mod p-1$ | $y=r^{m+r}\alpha^s \mod p$ |
| 15 | $sx=k+(m+r) \mod p-1$ | $y^s=r\,\alpha^{m+r} \mod p$ |
| 16 | $x=sk+(r+m) \mod p-1$ | $y=r^s\alpha^{m+r} \mod p$ |
| 17 | $(r+m)x=sk+1 \mod p-1$ | $y^{r+m}=r^s\alpha \mod p$ |
| 18 | $sx=(r+m)k+1 \mod p-1$ | $y^s=r^{r+m}\alpha \mod p$ |

**Table 1 18 ElGamal-Type Signature Variants**

## 3. Elliptic Curves over Finite Fields

*(a) Elliptic Curves over GF($2^n$)*

An non-supersingular elliptic curve E over *F* can be written into the following standard form

$$E: \quad y^2 + xy = x^3 + ax^2 + b,\ b \neq 0,\ a, b \in F.$$

The points *P = (x, y), x, y*$\in F$ that satisfy this equation, together with a ‘‘point at infinity” denoted *O* form an Abelian group (*E*, +, *O*) whose identity element is *O*. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two different points in *E* and both *P* and *Q* are not equal to the

infinity point. Addition Law for E non-supersingular: For $2P = P+P = (x_3, y_3)$ , if $x_1 \neq 0$,

$$x_3 = \delta^2 + \delta + a,$$

$$y_3 = (x_1 + x_3)\delta + x_3 + y_1,$$

where

$$\delta = x_1 + y_1 / x_1.$$

If $x_1 = 0$, $2P = O$.

For $P + Q = (x_3, y_3)$ , if $x_1 = x_2$ , then $P + Q = O$ . Otherwise,

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a,$$

$$y_3 = (x_1 + x_3)\lambda + x_3 + y_1,$$

where

$$\lambda = (y_1 + y_2) /(x_1 + x_2).$$

*(b) Elliptic curves over $K = GF(p^n)$, $p>2$.*

An non-supersingular elliptic curve $E$ over $K$ can be written into the following standard form

$$E: \quad y^2 = x^3 + ax + b,\ 4a^3 + 27b^2 \neq 0,\ a, b \in K.$$

We also have the addition law for the elliptic curve $E$ over $K$, please see [10].

## 4. EC- DSA Variants

In the following, we will generalize signature variants as listed in Table 1 to elliptic curve signature schemes. We will use the same setup as suggested in IEEE P1363/D4 standard form [11] for EC-DSA.

*a. System Generation:*

1. Choose $p$, a prime, and $n$, an integer, $f(x)$, an irreducible polynomial over $GF(p)$ of degree $n$, generating finite field $GF(p^n)$ with the defining polynomial $f(x)$, and assume that $\alpha$ is a root of $f(x)$ in $GF(p^n)$.
2. Generate a non-supersingualar curve E over $GF(p^n)$.
3. Choose a point $P = (x, y)$ on $E$ of order $q$ which is a prime.
4. Converting function:

$$c(x): GF(p^n) \rightarrow Z_{p^n}$$

which is given by

$$c(x) = \sum_{i=0}^{n-1} c_k p^i \in Z_{p^n}, \quad \text{for } x = \sum_{i=0}^{n-1} c_k \alpha^i \in GF(p^n),\ 0 \le c_i < p.$$

*b. Key Generation:*

1. Private key $d$, which is an integer, randomly selected as $0 < d < q$.
2. Public key $Q$, which is a point on $E$ and computed by $Q = dP = (x_d, y_d)$.

*c. Signing for a message m:*

1. Generate a one-time key pair $(k, R)$ in the following way: randomly *choose* $k: 0 < k < q$ and compute a point $R = kP = (x_k, y_k)$.
2. Compute $r$ in the following way: generating an integer by using the converting function for converting $x_k$ into a *p-ary* number:

$$x_k = \sum_{i=0}^{n-1} c_{i,k} \alpha^i \text{ converting into } c(x_k) = r = \sum_{i=0}^{n-1} c_{i,k} p^i, \quad 0 \le i < p.$$

3. Compute $s$:

$$s = dh(m) + kr \bmod q,$$

where $h(x)$ is the hashing function.

The pair $(r, s)$ is a signature of the message $m$.

*d. Verifying:*

1. Compute numbers:

$$t = r^{-1} \bmod q,$$
$$t_1 = ts \bmod q, \text{ and}$$
$$t_2 = h(m)t \bmod q.$$

2. Compute a point on $E$ by using the system key $P$ and user's public key $Q$:

$$t_1 P - t_2 Q = (x_e, y_e) \Rightarrow (r^{-1} s \bmod q)P - (h(m)r^{-1} \bmod q)\, Q = (x_e, y_e) \quad (A)$$

3. By using the converting function to compute the integer $c(x_e)$ and check if $r = c(x_e) \bmod q$. If this is true, then $(r, s)$ is accepted as a valid signature of the message $m$.

We list all 18 EC-DSA Harn's variants (EC-DSAH) in Table 2.

| equation number | signature equation | signature verification equation (A) |
|---|---|---|
| 1 | $h(m)d=rk+s \mod q$ | $(r^{-1}h(m) \mod q)Q - (r^{-1}s \mod q)P = (x_e, y_e)$ |
| 2 | $h(m)d=sk+r \mod q$ | $(s^{-1}h(m) \mod q)Q - (s^{-1}r \mod q)P = (x_e, y_e)$ |
| 3 | $rd=h(m)k+s \mod q$ | $(h(m)^{-1}r \mod q)Q - (h(m)^{-1}s \mod q)P = (x_e, y_e)$ |
| 4 | $rd=sk+h(m) \mod q$ | $(s^{-1}r \mod q)Q - (s^{-1}h(m) \mod q)P = (x_e, y_e)$ |
| 5 | $sd=rk+h(m) \mod q$ | $(r^{-1}s \mod q)Q - (r^{-1}h(m) \mod q)P = (x_e, y_e)$ |
| 6 | $sd=h(m)k+r \mod q$ | $(h(m)^{-1}s \mod q)Q - (h(m)^{-1}r \mod q)P = (x_e, y_e)$ |
| 7 | $rh(m)d=k+s \mod q$ | $(rh(m) \mod q)Q - (s \mod q)P = (x_e, y_e)$ |
| 8 | $d=h(m)rk+s \mod q$ | $(h(m)r)^{-1} \mod q)Q-((h(m)r)^{-1}s \mod q)P = (x_e, y_e)$ |
| 9 | $sd=k+h(m)r \mod q$ | $(s \mod q)Q-(h(m)r \mod q)P=(x_e, y_e)$ |
| 10 | $d=sk+rh(m) \mod q$ | $(s^{-1} \mod q)Q - (s^{-1}rh(m) \mod q)P = (x_e, y_e)$ |
| 11 | $rh(m)d=sk+1 \mod q$ | $(s^{-1}rh(m) \mod q)Q - (s^{-1} \mod q)P = (x_e, y_e)$ |
| 12 | $sd=rh(m)k+1 \mod q$ | $((h(m)r)^{-1}s \mod q)Q-((h(m)r)^{-1} \mod q)P = (x_e, y_e)$ |
| 13 | $(r+h(m))d=k+s \mod q$ | $((r+h(m)) \mod q)Q-(s \mod q)P=(x_e, y_e)$ |
| 14 | $d=(h(m)+r)k+s \mod q$ | $((h(m)+r)^{-1} \mod q)Q-((h(m)+r)^{-1}s \mod q)P= (x_e, y_e)$ |
| 15 | $sd=k+(h(m)+r) \mod q$ | $(s \mod q)Q-((h(m)+r) \mod q)P=(x_e, y_e)$ |
| 16 | $d=sk+(r+h(m)) \mod q$ | $(s^{-1} \mod q)Q - (s^{-1} (r+h(m)) \mod q)P = (x_e, y_e)$ |
| 17 | $(r+h(m))d=sk+1 \mod q$ | $(s^{-1} (r+h(m)) \mod q)Q - (s^{-1} \mod q)P = (x_e, y_e)$ |
| 18 | $sd=(r+h(m))k+1 \mod q$ | $((h(m)+r)^{-1}s \mod q)Q-((h(m)+r)^{-1} \mod q)P= (x_e, y_e)$ |

**Table 2. 18 EC-DSA Signature Variants**

## 5. Optimal Digital Signatures

Signing and verifying each EC-DSAH signature, in addition to evaluate points on the elliptic curve, require to compute modular inverses. Among above 18 EC-DSAH variants, (7), (9), (13) and (15), as listed in Table 2 do not need to compute both $k^1$ and $s^1$. Thus, we call these variants as the optimal EC-DSAH variants since these schemes require minimum amount of computations on both sides of applications.

## 6. Digital Signature with Broadband Subliminal Channel

Subliminal channel is a covert communication channel to send a message to an authorized receiver; but this message cannot be discovered by any unauthorized receiver. There are some applications that can take advantage of this by hiding secret messages in this subliminal channel. For example, a credit card provider can hide the cardholder's credit history and credit limit in a digital signature of a credit card.

Simmons [12] has observed that the ElGamal signature scheme requires $\beta$ bits of security against forgery and $\alpha$ bits are used to communicate a signature. Since $\alpha>\beta$, $\alpha-\beta$ bits are potentially available for subliminal communication. Simmons defined that if the subliminal channel uses all, or nearly all, of these $\alpha-\beta$ bits, it is said to be broadband; otherwise it is said to be narrowband.

In the EC-DSAH, the subliminal message is hidden in the parameter $k$. Since the subliminal receiver knows the secret key $d$, by knowing the public values of $\{r, s, m\}$, the subliminal receiver can extract the subliminal message $k$ according to the signature equation. However, since $q$ is a prime number, it is guaranteed that all modulo inverse *mod* $q$ do exist. Thus, all EC-DSAH are signature schemes with subliminal channel. In

particular, among above 18 EC-DSAH variants, (7), (9), (13) and (15), as listed in Table 2 have broadband subliminal channels since any subliminal message can be hidden in the parameter $k$.

## 7. Digital Multisignatures

Digital multisignature is signed by multiple signers with the knowledge of multiple secret keys and can be verified using all signers' public keys. An efficient digital multisignature scheme can combine multiple individual signatures of the same message into a single multisignature and this multisignature can be verified at once. The application of digital multisignature can be found in some secret sharing applications. For example, a company's policy may require multiple managers to sign business contracts together. Digital multisignature scheme enables this internal policy to be executed effectively. Each manager uses his individual secret key to produce an individual signature and all individual signatures can be combined into a single multisignature. However, to any external verifier, this multisignature is just a normal signature that can be verified by using the company's public key, which is a product of all public keys of the signers.

Harn [13, 14] has proposed two ElGamal-type variants that can combine all individual signatures into a multisignature without any data expansion. In other words, the length of the multisignature is equivalent to the length of each individual signature. The performance of Harn's multisignature schemes is reasonable since the length of the signature/multisignature depends only on the security parameters of signature schemes and does not depend on the number of signers involved. Multiple signers with knowledge of multiple secret keys can produce a fixed length of multisignature. Let us summarize properties associated with these multisignature schemes:

1. the length of multisignature is fixed;
2. the multisignature can be verified at once; and
3. the public key associated with this multisignature is just the product of all individual public keys.

Property (1) minimizes the communication and memory costs of multisignatures. Property (2) speeds up the verification process significantly. Property (3) reduces the size of public-key directory since it needs only to store each signer's public key.

Here, we would like to explore the design concept of the multisignature schemes [13, 14] and identify EC-DSAH variants in Table 2 that can provide similar function. We briefly review the multisignature scheme proposed in reference [14].

We assume that there are two signers, $U_1$ and $U_2$, to sign the same message $m$.

*a. Generating the Multisignature:*

*Phase 1: Determining the commitment value of r*

Each signer $U_i$ randomly selects a number $k_i$ from $[1, p-2]$ and computes

$r_i = \alpha^{k}i \bmod p,$

$\{ r_i\}$ is broadcast to the other signer. Once $r_1$ and $r_2$ are available through the broadcast channel, each signer computes the commitment value $r$ as

$r = r_1 r_2 \bmod p.$

*Phase 2: Determining the multisignature value of s*

Each signer uses his secret values, $x_i$ and $k_i$, to sign the message $m$. $U_i$ solves the

equation

$s_i = x_i m - k_i r \mod p-1$,

for integer $s_i$, where $0 \le s_i \le p-2$ and transmits $\{m, s_i\}$ to the clerk.

Once the clerk receives the individual signature $\{r_i, s_i\}$ from $U_i$, he needs to verify the validity of this individual signature. The verification procedure is to check

$y_i^m = r_i^r \alpha^s i \mod p$.

Once all individual signatures are received and verified by the clerk, the multisignature of message $m$ can be generated as $\{r, s\}$, where $s = s_1 + s_2 \mod p-1$.

*b. Verifying the Multisignature*:

Since individual signatures, $\{r_1, s_1\}$ and $\{r_2, s_2\}$ satisfy

$y_1^m = r_1^r \alpha^s 1 \mod p$, and

$y_2^m = r_2^r \alpha^s 2 \mod p$.

By multiplying these two equations, we obtain the multisignature verification equation as

$y^m = r^r \alpha^s \mod p$,

where $y = y_1 y_2 \mod p$.

***EC-DSAH multisignature schemes:***

We will generalize Harn's multisignature schemes to EC-DSAH variants.

*a. System Generation: the same as described in EC-DSA.*

*b. Determining the public keys: Assume there are v signers $U_I$,* for $1 \le I \le v$.

Each signer randomly selects an integer $d_i$ from $[1, q-1]$ and computes a corresponding public key as the point:

$$Q_i = d_i P = (x_{di}, y_{di}), \ 1 \le i \le v.$$

The public key, $Q$, associated with all signers is equivalent to the sum of all individual public keys, i.e.,

$$Q = Q_1 + Q_2 + \ldots + Q_v = dP = (x_d, y_d),$$

where $d = d_1 + d_2 + \ldots + d_v \mod q$.

*c. Generating the multisignature:*

*Phase 1: Determining the commitment value of r*

We assume that there are $v$ signers to sign the same message $m$. Each signer $U_i$ randomly selects a number $k_i$ from $[1, q-1]$ and computes

$$R_i = k_i P = (x_{ki}, y_{ki}), \text{ for } 1 \le i \le v,$$

and converting the x-coordinate into the integer $r_i = c(x_{ki})$ where $c(x)$ is the converting function. $r_i$ is broadcast to the other signer. Once $r_i$, $1 \le i \le v$, are available through the broadcast channel, each signer computes the commitment value $r$ as

$$r = r_1 + r_2 + \ldots + r_v \mod q.$$

*Phase 2: Determining the multisignature value of s*

Instead of signing the message *m* directly, all signers should sign the one-way hash result *m'=h(m)*, where *h* is the one-way hash function. Each signer uses his secret keys, $d_i$ and $k_i$, to sign the message *m'*. $U_i$ solves the equation

$$s_i = d_i m' + k_i r \mod q,$$

and transmits $(m, s_i)$ to the clerk.

Once the clerk receives the individual signature $(r_i, s_i)$ from $U_i$, he needs to verify the validity of this individual signature. The verification procedure is to compute

$$(r^{-1} s_i \mod q)P - (m' r^{-1} \mod q)Q_i = (x_{ei}, y_{ei}),\ 1 \le i \le v;$$

and check

$$r_i = c(x_{ei}) \mod q,\ 1 \le i \le v.$$

Once all individual signatures are received and verified by the clerk, the multisignature of message *m* can be generated as $(r, s)$, where

$$s = s_1 + s_2 + \ldots + s_v \mod q.$$

*d. Verifying the multisignature:*

Since individual signatures $(r_i, s_i)$, $1 \le I \le v$, satisfy

$$(r^{-1} s_i \mod q)P - (m' r^{-1} \mod q)Q_i = (x_{ei}, y_{ei}), 1 \le i \le v.$$

Adding the above equations from *1* through *v*, we obtain

$$(r^{-1} s \mod q)P - (m' r^{-1} \mod q)\, Q = (x_e, y_e),$$

where $s = s_1 + s_2 + \ldots + s_v \mod q$, $Q = Q_1 + Q_2 + \ldots + Q_v = dP = (x_d, y_d)$, and $r = c(x_e) \mod q$. In other words, Verifier computes the point $(x_e, y_e)$ and check if $r = c(x_e) \mod q$. If it is true, then $(r, s)$ is accepted as the valid multisignature of the message *m* signed by the users $U_1, \ldots U_v$.

EC-DSAH variants, (1), (3), (7), (8), (13) and (14), as listed in Table 2 can provide efficient multisignatures.

## 8. Batch Signing

Since generating digital signatures require intensive computations, it is desirable to speed up these computations by using either a special-purpose hardware or an efficient software algorithm. Batch signing allows a signer to sign messages for multiple recipients at once. Thus, it reduces the signature signing time significantly. Let us summarize the desirable properties of batch signing scheme:

1. batch signing scheme signs multiple messages at once;
2. batch signing signature can be verified by each recipient separately; and
3. unrelated messages should not be revealed to unrelated verifiers.

Property (1) ensures the advantage of using batch signing scheme. Property (2) ensures that to each recipient, the batch signing signature is the same as normal individual digital signature; but to the batch signature signer, he has signed multiple messages at once. Property (3) ensures the privacy of massages since unrelated messages should not be revealed to unrelated recipients. The batch signing scheme can be applied to applications, such as the payment gateway in the SET protocol, which requires to sign a large number of messages frequently.

The proposed algorithm can be applied to any digital signature scheme. We use the following example to illustrate this algorithm.

We assume that in a payment system the payment gateway (PG) needs to sign three different messages $m_1$, $m_2$, $m_3$, for three different merchants $M_1$, $M_2$, $M_3$, respectively. Let us denote $e$ as the public key of PG and $d$ as its private key, $Sign\{h(m)\}d$ as the signature of message $m$ using the private key $d$, and $h(m)$ as a one-way hash value of the message $m$, i.e., the message digest.

The PG does the following steps:

Step 1: Computes $h(m_1)$, $h(m_2)$, and $h(m_3)$.

Step 2: Computes $h(h(m_1)\perp h(m_2)\perp h(m_3))$, where "$\perp$" represents the data concatenation process.

Step 3: Computes $Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d$.

Step 4: Sends $\{ Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d, m_1, h(m_2), h(m_3)\}$ to $M_1$.
Sends $\{ Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d, m_2, h(m_1), h(m_3)\}$ to $M_2$.
Sends $\{ Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d, m_3, h(m_1), h(m_2)\}$ to $M_3$.

$M_1$ does the following steps after receiving $\{ Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d, m_1, h(m_2), h(m_3)\}$ from PG:

Step 1: Computes $h(m_1)$.

Step 2: By concatenating $h(m_1)$, $h(m_2)$, and $h(m_3)$ all together, computes $h(h(m_1)\perp h(m_2)\perp h(m_3))$.

Step 3: Using PG's public key $e$ and $h(h(m_1)\perp h(m_2)\perp h(m_3))$ obtained from Step 2, verifies the validity of the received signature $Sign\{ h(h(m_1)\perp h(m_2)\perp h(m_3))\}d$.

$M_2$ and $M_3$ can repeat the above steps to verify their own individual signature of corresponding message, respectively.

We have illustrated the method for the case of 3 merchants. Clearly, the method can be extended to the case of signing any number of $n$ messages simultaneously.

Although $h(m_2)$ and $h(m_3)$ have been sent to $M_1$ to enable $M_1$ to verify the signature of message $m_1$, but the contents of $m_2$ and $m_3$ have never been revealed to $M_1$. This satisfies the privacy requirement since sensitive messages should not be revealed to any unrelated receiver. On the other hand, to generate a valid signature for messages we still need the private key $d$. This ensures the security requirement of digital signature.

Instead of signing each individual message separately, our algorithm enables the payment gateway to sign three messages simultaneously. In general, the time for computing hash function is negligible compared to the time for computing signature. Thus, our algorithm reduces the computational time by a factor of three. There is a trade-off between computational time and the communication cost. Since two additional hash values need to be included in the normal signature, the cost is 320 bits if SHA-1 hash [15] is used. Further saving in the computational time can be achieved if the payment gateway signs $n$ messages simultaneously. But, longer hash values need to be sent over the network.

## 9. **Batch Verifications**

Batch verification allows a verifier to verify multiple signatures signed by the same signer at once. Thus, it reduces the signature verification time significantly. Let us summarize the desirable properties of batch verification:

1. multiple signatures can be verified at once; and
2. time for batch verification should be almost constant.

The application of batch verification can be found in some traffic congested gateways that require to verify X. 509 public-key certificates [16] signed by the same Certificate Authority (CA).

Naccache et al. [17] proposed an interactive DSA batch verification protocol, that the signer generates $t$ signatures through interactions with the verifier, and then the verifier validates these $t$ signatures at once based on one batch verification criterion. Lim and Lee [18] pointed out that the interactive DSA batch protocol proposed by Naccache et al. is insecure. Recently, Harn [19] proposed secure DSA-type batch verification algorithms.

*Secure DSA-type batch verification*

We assume that DSA-type algorithms share the same parameters and similar signing and verification forms as the original DSA algorithm. The following DSA-type digital signature algorithm is based on one of 18 ElGamal-type digital signature schemes developed in reference [8]. For each message $m$ to be signed a new random integer $k$ from $[1,q-1]$ is privately selected and then $r$ is computed as $r=(g^k \bmod p) \bmod q$. With the knowledge of the secret key $x$ the signer solves $s$ that satisfies $s=rk-mx \bmod q$. $\{r, s\}$ is the signature of $m$. With the knowledge of signer's public key $y$, the signature can be verified by checking whether that $r=(g^{sr'} y^{mr'} \bmod p) \bmod q$, where $r'=r^{-1} \bmod q$.

Let us assume that there are $t$ signatures $\{r_1, s_1\}, \{r_2, s_2\},\ldots, \{r_t, s_t\}$ of $t$ different messages $m_1, m_2,\ldots, m_t$, respectively. These $t$ signatures satisfy the following $t$ equations as

$$r_1=(g^{s_1 r_1'} y^{m_1 r_1'} \bmod p) \bmod q,$$
$$r_2=(g^{s_2 r_2'} y^{m_2 r_2'} \bmod p) \bmod q,$$

$$\vdots$$

$$r_t=(g^{s_t r_t'} y^{m_t r_t'} \bmod p) \bmod q.$$

By multiplying these $t$ equations together, we obtain the batch verification criterion as

$$r_1 r_2 \ldots r_t = (g^{s_1 r_1'+s_2 r_2'+\ldots+ s_t r_t'} y^{m_1 r_1'+m_2 r_2'+\ldots+ m_t r_t'} \bmod p) \bmod q.$$

We claim that the verifier can verify these $t$ signatures simultaneously by checking this batch verification criterion. If both sides of the equation are identical, the $t$ signatures $\{r_1, s_1\}, \{r_2, s_2\},\ldots, \{r_t, s_t\}$ of messages $m_1, m_2,\ldots, m_t$, can be verified. However, if the identity does not hold, where this may results from either transmission noise or invalid individual signatures, we need to verify each individual signature separately. It is obvious that to verify these $t$ signatures according to the batch verification criterion requires 2 modular exponentiations. However, if the verifier verifies each individual signature separately, it requires $2t$ modular exponentiations.

We would like to point out that the Lim and Lee's attack does not work properly in this proposed algorithm. In their attack, the attacker can randomly select all $r_i$

first. Then the attacker can solve $s_i$ accordingly. However, in our proposed algorithm $r_i$ cannot be randomly selected at the first place. On the other hand, the signature algorithm used to sign each individual signature is also secure.

***EC-DSAH Batch Verification***

Here, we will give secure EC-DSAH batch verification algorithms. The signer signs message $m$ by computing $s= m'd+kr \mod q$ and $\{r, s\}$ is the signature of $m$.

Let us assume that there are $v$ signatures $\{r_1, s_1\}, \{r_2, s_2\},\ldots, \{r_v, s_v\}$ of $v$ different messages $m_1, m_2,\ldots, m_v$, respectively. These $v$ signatures satisfy the following $v$ pairs of equations as

$$( r_i^{-1}s_i \mod q)P - (m_i'r_i^{-1} \mod q)\ Q= R_i= (x_{ri}, y_{ri}),\ 1\le i \le v;$$

and

$$r_i = c(x_{ri}) \mod q,\ 1\le i \le v.$$

By adding these $v$ equations together, we obtain the batch verification criterion as

$$\left(\sum_{i=1}^{v} r_i^{-1}s_i \bmod q\right)P - \left(\sum_{i=1}^{v} r_i^{-1}m_i' \bmod q\right)Q = (x_r, y_r);$$

and

$$\sum_{i=1}^{v} c(x_i) = c(x_r).$$

We claim that the verifier can verify these $v$ signatures simultaneously by checking the above two batch verification equations. I.e., computing the $x$-coordinate $x_r$, and checking if both sides of the above equation are equal. If so, the $v$ signatures $\{r_1, s_1\}, \{r_2, s_2\},\ldots, \{r_v, s_v\}$ of messages $m_1, m_2,\ldots, v_t$, can be verified. However, if the identity does not hold, where this may results from either transmission noise or invalid individual.

Actually, the EC-DSAH signature variants, (1), (3), (5), (6), (7), (8), (9), (12), (13), (14), (15) and (18), as listed in Table 2 can also be used to develop similar batch verification criteria.

## 10. Conclusion

We have introduced elliptic curve signature variants and accessories associated with these variants. Each accessory has one unique feature. Based on these features, additional services can be provided through normal digital signatures. Our discussion is focused on the 18 EC-DSAH variants.

There are two researches that are currently under investigation. First, other accessories, such as blind signature and undeniable signature, which related to EC-DSAH variants will be investigated. Second, the methods to establish RSA-type accessories, such as multisignature and subliminal channel, are still remained as open problems.

**References**


[1] D. E. Denning. Cryptography and Data Security. Addison-Wesley, Reading, MA, 1982.

[2] R. L. Rivest, A. Shamir and L. Adelman. A method for obtaining digital signatures and public-key cryptosystem. In *Commun. Of ACM,* Vol. 21, No. 2, pp. 120-126, Feb. 1978.

[3] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Trans. Inform. Theory,* Vol. IT-31, pp. 469-472, July, 1985.

[4] Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). In *Federal Register,* Vol. 56, No. 169, 30 Aug. 1991, pp. 42980-42982.

[5] SET Specification. http://www.visa.com/cgi-bin/vee/ht/ecomm/set/downloads. html?2+0#specs.

[6] The Secure Sockets Layer Protocol. http://www.netscape.com/info/security-doc.html.

[7] The Resolution of ISAKMP with Oakley. ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipsec-isakmp-oakley-04.txt.

[8] L. Harn and Y. Xu. Design of Generalized ElGamal type digital signature schemes based on discrete logarithm. In *Electronics Letters*, Vol. 30, No. 24, Nov. 1994, pp. 2025-2026.

[9] P. Horster, H. Petersen, and M. Michels. Meta-ElGamal signature schemes. In *Proceedings of the 2nd Annual ACM Conference on Computer and Communications Security,* ACM Press, 1994, pp. 96-107.

[10] A.J. Menezes. Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, 1993.

[11] IEEE P1363/D4: Standard Specifications for Public Key Cryptography, *the Institute of Electrical and Elctronics Engineers, Inc.* , June 16, 1998.

[12] G. J. Simmons. Subliminal Communication is Easy Using the DSA. Presented at *Eurocrypt '93, Lofthus*, Norway, 1993.

[13] L. Harn. A New Digital Signature Based on the Discrete Logarithm. *In Electronics Letters*, Vol. 30, No. 5, March 1994, pp. 193-195.

[14] L. Harn. Group-oriented (*t, n*) Threshold Signature and Multisignature. In *IEE Proceedings-Computers and Digital Techniques,* Vol. 141, No. 5, Sep. 1994, pp. 307-313.

[15] National Institute of Standards and Technology. Secure hash standard. In *NIST FIPS PUB 180,* U. S. Department of Commerce, DRAFT, 1993.

[16] CCITT, Recommendation X.509. The Directory-Authentication Framework. Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989.

[17] D. Naccache, D. M'Raihi, D. Rapheali, and S. Vaudenay. Can DSA be Improved: Complexity Trade-Offs with the Digital Signature Standard. In *Pre-Proceedings of Eurocrypt '94*, pp. 85-94.

[18] C. H. Lim and P. J. Lee. Security of Interactive DSA Batch Verification. In *Electronics Letters*, Sep. 1994, Vol. 30, No. 19, pp. 1592-1593.

[19] L. Harn. Batch Verifying Multiple DSA-type Digital Signatures. In *Electronics Letters*, Vol. 34, No. 9, April, 1998, pp. 870-871.