Atanasiu A. A new batch verifying scheme for identifying illegal signatures. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 28(1): 144–151 Jan. 2013. DOI 10.1007/s11390-013-1318-4

A New Batch Verifying Scheme for Identifying Illegal Signatures

Adrian Atanasiu

Faculty of Mathematics and Computer Science, Bucharest University, Str. Academiei 14, Bucharest 010014, Romania

E-mail: aadrian@gmail.com

Received March 12, 2012; revised September 3, 2012.

Abstract The concept of batch verifying multiple digital signatures is to find a method by which multiple digital signatures can be verified simultaneously in a lower time complexity than separately verifying all the signatures. In this article, we analyze the complexity of the batch verifying schemes defined by Li, Hwang and Chen in 2010, and propose a new batch verifying multiple digital signature scheme, in two variants: one for RSA – by completing the Harn's schema with an identifying illegal signatures algorithm, and the other adapted for a modified Elliptic Curve Digital Signature Algorithm protocol.

Keywords digital signature, batch verifying, multiple signature, RSA signature, elliptic curve digital signature algorithm standard

1 Introduction

In a signature protocol, a user generates a set of documents $m_1, \ldots, m_p$ and signs them $(sig_K(m_1), \ldots, sig_K(m_p))$ using a private key K; finally the set $\{(m_i, sig_K(m_i) \mid 1 \leqslant i \leqslant p\}$ is transmitted to another user. This receiver finds the signer's public key and verifies each of these p signatures one-by-one. This authentication and verification phase usually takes a large amount of computation time.

The batch verifying multiple digital signatures is a method that can efficiently improve the performance of verifying multiple digital signatures in a single step. It can be completed with batch identification – a method to find invalid signatures if a set of signatures contains invalid signature. These directions of research: batch verification[1-5] and batch identification[6-9], are completed for some schemes with a security analysis[10-15].

The importance of the subject can be pointed out by the vast domain of proposed schemes: there are schemes for short signatures[3,16], identity-based signatures[17-19], group signatures[7], proxy signatures[20], signatures based on elliptic curves[9,11], or for mobile devices[21], etc. The majority of the proposed schemes consider RSA signatures[6,8,12,15,22-24], or DSA signatures[13,25].

In this paper we will consider an idea developed by Harn L.[24] in 1998 for verifying multiple RSA signatures simultaneously in only one exponential operation. Some weaknesses of this method are observed in [12], but the main problem of Harn's protocol is that the verifier can detects the existence of illegal signatures but he/she cannot detect which signatures fail. In 2010, Li, Hwang and Chen[26] have proposed a matrix-based solution (LHC scheme) which can give some information regarding the localization of illegal signatures. But, no matter what the size of the set of signed messages is, only one illegal signature can be surely identified.

Our contribution consists in an analysis of the complexity for hyper-cube generalized LHC scheme. The conclusion is that the original LHC algorithm is optimal from this point of view. Then we complete the verification of Harn's schema with a divide-and-impera type algorithm for detecting the illegal signatures. Using this algorithm, if up to 10% of the signatures are illegal, then they can be completely identified. Moreover, the case of illegally compensated messages is considered. Finally, a proposal for applying this algorithm to a modified Elliptic Curve Digital Signature Algorithm (ECDSA) scheme completes our construction.

Therefore the paper has two main parts. The first one is dedicated to an analysis of the complexity and the number of illegal signatures which can be detected by normal and extended LHC batch verifying schemes. The second part improves these results by proposing a simple batch verifying scheme for detecting and identifying illegal signatures. This algorithm is built in two variants: for RSA signatures, and for a modified form of ECDSA standard.

The paper is organized as follows. Section 2 is dedicated to basic notions and terms used in the paper.

Regular Paper

In Section 3 the Harn and LHC schemes (original and extended) are presented and a comparative analysis of their complexity is developed. Section 4 proposes a new simple scheme for verifying RSA batch signature. This algorithm is adapted in Section 5 for a modified ECDSA standard. The last section concludes the paper.

2 Formal Preliminaries

2.1 Cryptographic Hash Function

A cryptographic hash function is used in security area in order to provide assurance of data integrity.

Definition 1. *A hash function is a triple* $(\mathcal{X}, \mathcal{Y}, h)$, *where*:

1) $\mathcal{X}$ *is a set of messages,*

2) $\mathcal{Y}$ *is a set of messages digests,*

3) $h : \mathcal{X} \longrightarrow \mathcal{Y}$ *is a surjective mapping.*

A hash function is cryptographic if it is infeasible to find two values $x, y \in \mathcal{X}$ $(x \neq y)$, *such that* $h(x) = h(y)$.

It is easy to see[27] that a cryptographic hash function is one-way (knowing $y \in \mathcal{Y}$ it is infeasible to find $x \in \mathcal{X}$ such that $y = f(x)$).

2.2 Signature Schemes

Definition 2[27]**.** *A signature scheme is a five-tuple* $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ *where*:

• $\mathcal{P}, \mathcal{A}, \mathcal{K}$ *are finite sets of messages, signatures, and keys respectively,*

• *For each* $K \in \mathcal{K}$, *there is a signing algorithm* $sig_K \in \mathcal{S}$ *and a corresponding verification algorithm* $ver_K \in \mathcal{V}$. *Each* $sig_K : \mathcal{P} \longrightarrow \mathcal{A}$ *and* $ver_K : \mathcal{P} \times \mathcal{A} \longrightarrow \{true, false\}$ *are functions such that*

$$ver_K(x, y) = true \Leftrightarrow y = sig_K(x).$$

A pair $(x, y) \in \mathcal{P} \times \mathcal{A}$ *is called a signed message.*

The two signature schemes used in this paper are RSA and ECDSA.

2.2.1 RSA Signature Scheme

• *Setup.* Let $n = p \times q$, where p and q are primes. We define $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$, $\mathcal{K} = \{(n, p, q, d, e) \mid n = p \times q,\ p, q \text{ are primes}, d \times e \equiv 1 \pmod{\phi(n)}\}$, where $\phi(n) = (p-1)(q-1)$ is the Euler function.

The values (n, e) are the public keys, and (p, q, d) represent the private keys.

To avoid some direct attacks, a cryptographic hash function $h : \mathbb{Z}_n \to \mathbb{Z}_n$ is used.

• *Signing Algorithm.* For $K = (n, p, q, d, e)$, define $sig_K(m) = [h(m)]^d \pmod{n}$.

• *Verification Algorithm.*

$$ver_K(m, s) = \text{true} \Leftrightarrow s^e \equiv h(m) \pmod{n}.$$

2.2.2 ECDSA Scheme

In 2000, the ECDSA was approved as Federal Information Processing Standard (FIPS) 186-2. The details of this scheme are presented in Section 5.

2.3 Elliptic Curves

There are many references which detail elliptic curve notions. In this paper we shall use definitions and results presented in [28] and [29].

Let q be a large prime. An elliptic curve E over $\mathbb{Z}_q$ is defined by

$$E:\ Y^2 \equiv X^3 + aX + b \pmod{q},$$

where $a, b \in \mathbb{Z}_q$, and $4a^3 + 27b^2 \neq 0$. The set

$$E(L) = \{(x, y) \in L \times L \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

of rational points of E, where L is an extension of $\mathbb{Z}_q$ and $\mathcal{O}$ is the point of infinity, can be structured as an abelian group with an additive composition law. If $P \in E(L)$ we denote by $\langle P \rangle = \{kP \mid k \geqslant 1\}$ the additive subgroup of $E(L)$ generated by P.

Definition 3. *Let* n *be a positive integer and* G_1, G_2 *be abelian groups written in additive notation, where* $nP = 0$ *for any* $P \in G_1 \cup G_2$. *Let* G_3 *be a cyclic group of order* n *written in multiplicative notation having* 1 *as identity. A bilinear pairing is a function* $e : G_1 \times G_2 \longrightarrow G_3$ *with the properties*

• $(\forall P, P' \in G_1,\ \forall Q, Q' \in G_2)$
$e(P + P', Q) = e(P, Q) \times e(P', Q)$,
$e(P, Q + Q') = e(P, Q) \times e(P, Q')$;

• $\forall P \in G_1 \setminus \{0\}, \exists Q \in G_2$ *with* $e(P, Q) \neq 1$;

• $\forall Q \in G_2 \setminus \{0\}, \exists P \in G_1$ *with* $e(P, Q) \neq 1$.

The importance of bilinear pairing in security information area is permanently growing. Almost all theoretical protocols based on elliptic curves use at least one bilinear pairing (especially Tate pairings and Weil pairings).

3 Batch Verifying Schemes

3.1 Li-Hwang-Chen Algorithm

Let us consider the set of p signed messages $\mathcal{MS} = \{(m_i, s_i) \mid 1 \leqslant i \leqslant p\}$, and h be a cryptographic hash function.

One of the first batch schemes presented in literature which concerns RSA signature verification was proposed by Harn[24]:

$$\Big[\prod_{t=1}^{p} s_t\Big]^e = \prod_{t=1}^{p} h(m_t) \pmod{n}. \tag{1}$$

Its advantage is that in only one exponentiation can be decided if there are illegal signatures. When (1) is verified, then all the messages from $\mathcal{MS}$ are authenticated.

The disadvantage appears when (1) fails. It means that there are illegal signatures, but they cannot be identified; moreover, their number is unknown. The next step which follows the scheme will be the signature-by-signature verification of all the messages from $\mathcal{MS}$ (therefore p more exponentiations have to be accomplished).

In 2010, Li, Hwan and Chen[26] proposed another algorithm (LHC scheme). The signatures of messages from $\mathcal{MS}$ are distributed (using a fixed permutation) in an $r \times s$ array (with $|r-s|$ minimum). (1) is replaced by $r+s$ similar relations, one for each row and column of the array.

The advantage of LHC scheme consists in a reduced number of computations (exponentiations and multiplications).

The weakness of LHC scheme is that it can detect but it cannot identify multiple illegal signatures. For example let us consider a batch of 16 signed messages $\{(m_i, s_i) \mid 0 \leqslant i \leqslant 15\}$ arranged in the 4×4 array

$$\begin{array}{cccc} S_{1,1} & S_{1,2} & \boldsymbol{S}_{1,3} & S_{1,4} \\ \boldsymbol{S}_{2,1} & S_{2,2} & S_{2,3} & S_{2,4} \\ S_{3,1} & S_{3,2} & S_{3,3} & \boldsymbol{S}_{3,4} \\ S_{4,1} & \boldsymbol{S}_{4,2} & S_{4,3} & S_{4,4} \end{array}$$

where $s_i = S_{\lfloor i/4 \rfloor + 1, i \pmod 4 + 1}$ and bold entries are illegal signatures. All check relations defined by the LHC scheme fail, without being able to decide which are the illegal signatures precisely.

More recently unpublished results extend this algorithm to a c-dimensional (c-D) hyper-cube (the LHC scheme being a variant in the case $c = 2$).

Let us analyze the advantages and disadvantages of such construction.

3.1.1 Complexity of Extended LHC Scheme

Let k, c be two positive integers such that $p \leqslant k^c$ and the p signatures of messages be arranged — using a fixed permutation — as entries of a c-D hyper-cube $S = [S_{i_1,\ldots,i_c}](1 \leqslant i_1, \ldots, i_c \leqslant k)$.

Therefore for each $(m_t, s_t) \in \mathcal{MS}$ there exists in S a unique point $(i_1, \ldots, i_c)$ with $S_{i_1,\ldots,i_c} = s_t$.

If $p < k^c$, then there will exist free entries in the hyper-cube; they are initialized by 1.

Using this arrangement, $c \times k^{c-1}$ relations will be constructed as follows:

Let j $(1 \leqslant j \leqslant c)$ be an arbitrary index. If (m_{t_r}, s_{t_r}) $(1 \leqslant r \leqslant k)$ are k signed messages from $\mathcal{MS}$ with $S_{i_1,\ldots,i_{j-1},r,i_{j+1},\ldots,i_c} = s_{t_r}$, then the r-th equation of verification defined on the j-coordinate is (computations mod n)

$$\Big[\prod_{r=1}^{k} S_{i_1,\ldots,i_{j-1},r,i_{j+1},\ldots,i_c}\Big]^e = \prod_{r=1}^{k} h(m_{t_r}). \qquad (2)$$

All these relations use in total $c \times k^{c-1}$ exponentiations and $2(k-1) \times c \times k^{c-1}$ multiplications.

If we denote by C_e the time complexity of one exponentiation and C_m the time complexity of one multiplication, the extended LHC scheme is better than the signature-by-signature verification of messages from $\mathcal{MS}$ if the following relation holds:

$$C_e \times c \times k^{c-1} + 2c \times (k-1) \times k^{c-1} < C_e \times k^c$$

or

$$(k-c) \times C_e > 2c \times (k-1) \times C_m.$$

For $k \leqslant c$ this inequality is not true (hence the batch LHC scheme is not recommended).

For $k > c$ we obtain

$$C_e/C_m > \frac{2c(k-1)}{k-c}. \qquad (3)$$

Let us denote $\alpha = C_e/C_m$ (the value of α is a constant which depends on the hardware architecture and the operation system); then (3) can be written as

$$k(\alpha - 2c) > c(\alpha - 2). \qquad (4)$$

Because the number of (2) (and therefore the number of exponentiations) is proportional with c, the most convenient choice — for large values of p — is $c = 2$; then $k > 2 + 4/(\alpha - 4)$, an inequality which can be easily verified by the parameters of the scheme (in general α takes values of at least two digits).

Example 1. Let us consider a batch of $p = 1\,000\,000$ signed messages. It can be associated with various hyper-cubes k^c where $k > c \geqslant 2$. Table 1 shows, for each case, the number E of exponentiations and the number M of multiplications.

Table 1.

No.	k	c	E	M	Cost	Verification of (4)
1	10	6	600 000	10 800 000	16 800 000	NO
2	16	5	327 680	9 830 400	13 107 220	NO
3	32	4	131 072	8 126 464	9 437 184	YES
4	100	3	30 000	5 940 000	6 240 000	YES
5	1 000	2	2 000	3 996 000	4 016 000	YES

The last two columns give an evaluation of the cost of verifications for the cases $\alpha = 10$ and (4) is verified respectively.

As a remark, the 4- and 5-D hyper-cubes (cases 2 and 3) have a great number of computations, because they contain many unused entries.

The cost of verification one-by-one of all $p = 1\,000\,000$ signatures is $10 \times 10^6 = 10\,000\,000$. From Table 1 the last two cases are recommended from the complexity point of view; the first two cases are eliminated because they do not fulfill the restriction (4), while cases 2 and 3 verify a larger batch of signatures (although many of them are dummies).

Conclusion. Although there are situations when the arrangement of signatures in a k^c hyper-cube with $c \geqslant 3$ is advantageous, asymptotically the original LHC scheme offers a lower time complexity than the extended version.

3.1.2 Number of Illegal Signatures Correctly Identified

Let us study the capacity of the extended LHC scheme to uniquely identify (without other punctual verifications) the illegal signatures.

Theorem 1. *Let us consider a c-D hyper-cube with edge k which contains all the signatures from $\mathcal{MS}$, and let t be the number of illegal signed messages.*

1) *If $t \leqslant c-1$, then all illegal signatures can be identified.*

2) *If $c \leqslant t \leqslant k^{c-1}$ then the identification of illegal signatures is possible only for convenient arrangements of signatures inside of the hyper-cube.*

3) *If $t > k^{c-1}$ then the existence of illegal signatures is signaled, but their identification is possible only one-by-one.*

Proof. From the modality of representing the signatures in the hyper-cube, some remarks arise:

• Relation (2) represents a line in the hyper-cube parallel with one of the c-D axis of coordinates.

• A signature is completely identified if it is at the intersection of exactly c lines.

If $t = 1$, one illegal signature is always identified: it invalidates c and only c relations of type (2); namely that c lines which contain the point associated with this signature; the intersection of sets of signatures associated with each line contains only one element: the signature which invalidates these c relations.

Because the formalism for a complete proof is very complex and the details take a lot of space, we will point out only some parts of the justifications for cases $c = 2$ and $c = 3$.

• $c = 2$. One error is completely identified if two relations of (2) fail; their locations give the row and column where the illegal signature is stored.

Two errors cannot be always found. For example, if S_{i_1,j_1} and S_{i_2,j_2} are two illegal signatures and $i_1 \neq i_2$, $j_1 \neq j_2$, then four relations will be invalided, corresponding to rows i_1, i_2 and columns j_1, j_2. The intersections two-by-two of these lines give $\{S_{i_1,j_1}, S_{i_1,j_2}, S_{i_2,j_1}, S_{i_2,j_2}\}$ as illegal signatures. That is not true, because two of them are correct.

The maximal number of illegal signatures which can be identified is k. This is possible if all those wrong signatures are on a single line; they will modify $k+1$ relations (2): the equation of that line and the equations of all columns. These assertions remain true if terms "line" and "column" are switched.

• $c = 3$. The existence of two illegal signatures affects five (if they are in the same equation) or six relations — two for each axis of the cube. Each intersection of three sets of signatures, each of which corresponding to a line parallel with one of the axes, will have at most one element.

Three errors cannot be always identified. For example, let us consider the illegal signatures $S_{i,j,k}, S_{i+2,j,k+2}, S_{i+2,j+2,k}$ situated in the cube in points P, Q and R respectively. They will affect nine relations, each group of three equations of type (2) corresponding to lines parallel with a coordinate axis Ox, Oy and Oz respectively.

We shall denote

$$
\begin{aligned}
P_x &= \{\ldots, S_{i,j,k}, S_{i+1,j,k}, S_{i+2,j,k}, \ldots\},\\
P_y &= \{\ldots, S_{i,j,k}, S_{i,j+1,k}, S_{i,j+2,k}, \ldots\},\\
P_z &= \{\ldots, S_{i,j,k}, S_{i,j,k+1}, S_{i,j,k+2}, \ldots\},\\
Q_x &= \{\ldots, S_{i,j,k+2}, S_{i+1,j,k+2}, S_{i+2,j,k+2}, \ldots\},\\
Q_y &= \{\ldots, S_{i+2,j,k+2}, S_{i+2,j+1,k+2}, S_{i+2,j+2,k+2}, \ldots\},\\
Q_z &= \{\ldots, S_{i+2,j,k}, S_{i+2,j,k+1}, S_{i+2,j,k+2}, \ldots\},\\
R_x &= \{\ldots, S_{i,j+2,k}, S_{i+1,j+2,k}, S_{i+2,j+2,k}, \ldots\},\\
R_y &= \{\ldots, S_{i+2,j,k}, S_{i+2,j+1,k}, S_{i+2,j+2,k}, \ldots\},\\
R_z &= \{\ldots, S_{i+2,j+2,k}, S_{i+2,j+2,k+1}, S_{i+2,j+2,k+2}, \ldots\},
\end{aligned}
$$

the sets of signatures contained by these relations. Then $P_x \cap P_y \cap P_z = \{S_{i,j,k}\}$, $Q_x \cap Q_y \cap Q_z = \{S_{i+2,j,k+2}\}$, $R_x \cap R_y \cap R_z = \{S_{i+2,j+2,k}\}$; in addition, it appears $P_x \cap Q_z \cap R_y = \{S_{i+2,j,k}\}$ as a new possible illegal signature.

The maximal number of illegal signatures which can be identified is k^2; this is possible when all these objects are situated in the same plane. Then, all $2k$ relations defined by this plane, and k^2 relations of lines perpendicular on it are affected, while the other $2k^2 - 2k$ relations corresponding to other $k-1$ planes remain satisfied. □

Obviously, for $n \leqslant t \leqslant k^{n-1}$ different criteria of identifying illegal signatures can be constructed, using geometrical properties of hyper-cubes and certain arrangements of signatures.

As a final conclusion, the number of illegal signatures surely identified (randomly localized in the hyper-cube) is proportional with n, and their number is very small in comparison with the number of signed messages from $\mathcal{MS}$.

4 New Algorithm for Batch Verification of RSA Signatures

Let us consider the RSA signature protocol, defined (in Subsection 2.2.1) by the public verification key (n, e) and the private signature key (p, q, d). Let

$$\mathcal{MS} = \{(m_i, s_i) \mid 1 \leqslant i \leqslant p\}$$

be a set of p signed messages, ordered by a total order relation. For example

$$(m_i, s_i) < (m_j, s_j) \Leftrightarrow (s_i < s_j) \vee [(s_i = s_j) \wedge (m_i < m_j)].$$

Algorithm 1 identifies an illegal signature (if exist) in the batch $\mathcal{MS}$ (all computations are performed mod n).

Algorithm 1.

Input: $\mathcal{MS}$, h cryptographic hash function.

1. $X = \prod_{t=1}^{p} s_t$, $Y = \prod_{t=1}^{p} h(m_t)$;
2. **If** $X^e = Y$ then **Return** "No error"
   else
   a) $low \leftarrow 1$, $high \leftarrow p$;
   b) $mid \leftarrow \lfloor (low + \text{high})/2 \rfloor$;
   c) $X = s_{low} \dots s_{mid}$, $Y = h(m_{low}) \dots h(m_{mid})$;
   d) **If** $X^e = Y$ **then** $low \leftarrow mid + 1$
   else $high \leftarrow mid$;
   e) **If** $low < high$ **then goto** 2(b);
   f) **Return**: s_{low} "is an illegal signature for" m_{low}.
3. **Stop**.

The scheme uses Harn's formula for detecting possible illegal signatures. If this formula fails, with a bisection algorithm, in $\lceil \log_2 p \rceil$ steps an illegal signed message will be identified.

Example 2. If $p = 1\,000$, then 1 000 exponentiations will be necessary for checking all signatures one-by-one. With Algorithm 1 only 10 exponentiations are enough for identifying one illegal signature.

The number of multiplications used is $(p-1)\big[1 + \frac{1}{2} + \frac{1}{2^2} + \cdots\big] \approx 2(p-1)$.

If the goal is to find all illegal signatures from the batch $\mathcal{MS}$, then two possibilities can be pursued:

• $\mathcal{MS} := \mathcal{MS} \setminus \{(m_{low}, s_{low})\}$ and Algorithm 1 is running again.

• The signed message (m_{low}, s_{low}) is replaced by $(1, 1)$ in $\mathcal{MS}$, and Algorithm 1 is running again.

4.1 Analysis of Complexity

Let us consider the first variant of finding the illegal signatures. If there are t $(0 \leqslant t \leqslant p)$ messages with wrong signatures in the batch $\mathcal{MS}$, then they are all identified completely after approximatively $\lceil \log_2 p + \log_2(p-1) + \log_2(p-t+1) \rceil = \lceil \log_2(t!(\frac{p}{t})) \rceil$ exponentiations and $(p-t) + \cdots + (p-1) = 2p - t - 1$ multiplications.

Like before, let us denote by C_e the time complexity of one exponentiation and C_m the time complexity of one multiplication.

Because a signature-by-signature verification of messages will spend p exponentiations, then t applications of Algorithm 1 which can find one illegal signature at each running will be preferable if

$$C_e \times \left\lceil \log_2 \left(t! \Big(\frac{p}{t} \Big) \right) \right\rceil + C_m \times (2p - t - 1) < p \times C_e$$

or

$$\frac{C_e}{C_m} > \frac{2p - t - 1}{p - \left\lceil \log_2 \left(t! \Big(\frac{p}{t} \Big) \right) \right\rceil}. \tag{5}$$

Example 3. Let us consider a batch of $p = 1\,000$ signed messages. Table 2 shows the complexity rates when Algorithm 1 is preferable, in the case of identifying t illegal signatures.

Table 2.

t	1	5	50	75	100	105	107
C_e/C_m	> 2.016	> 2.026	> 3.6	> 6.4	> 14.3	> 80.6	> 386

As a remark, if these 1 000 messages contain more than 108 illegal signatures, then Algorithm 1 becomes inefficient (versus one-by-one checking of signatures).

Obviously, (5) is functional only when $p - \lceil \log_2(t!(\frac{p}{t})) \rceil > 0$, or

$$2^p > t! \Big(\frac{p}{t} \Big). \tag{6}$$

Remark 1.

• *For the detection and deletion of one illegal signed message, Algorithm* 1 *is optimal — versus the Harn's and LHC's schemes*[12,24,26]*. For* $t > 1$*, the efficiency of Algorithm* 1 *is directly proportional with the values of* p*. So, for* $p = 10$*, at most two illegal signatures can be efficiently identified, whilst for* $p = 1\,000$ *the number of illegal signatures identified is greater than* 100*.*

• *We point out once more that the above remark concerns the efficiency of Algorithm* 1 *versus the classical method of separately checking the authenticity of each signature. If we ignore the idea of complexity, any number of illegally signed messages from* $\mathcal{MS}$ *can be identified and then avoided, using this algorithm.*

4.2 Case of Illegally Compensated Messages

It is possible that the batch $\mathcal{MS}$ contains illegally signed messages, despite step 2 of Algorithm 1 being verified. This is the case when the errors which appear are compensated, the final result obtained being correct. When the signed messages are arbitrarily generated, this situation has a negligible probability, and it can be ignored.

But, if h is not a cryptographic hash function, and there is an algorithm $\mathcal{A}$ which finds $x = \mathcal{A}(h(x))$ in polynomial time, then it is possible for an illegal sender to put in $\mathcal{MS}$ a set of illegally signed messages (r_i, s_i) with $s_i^e \neq h(r_i)$ $(1 \leqslant i \leqslant v+1)$ authenticated by relation (1). The intruder will proceed as follows:

1) Computes $A = \prod_{t=1}^{v} h(r_t) \pmod n$;

2) Finds (using the Euclid Generalized Algorithm) B so that $A \times B \equiv 1 \pmod n$;

3) Applies the algorithm $\mathcal{A}$ and finds the message $r_{v+1} = \mathcal{A}(B)$;

4) Generates randomly $s_1, \ldots, s_v$ and chooses s_{v+1} which verify $\prod_{t=1}^{v+1} s_t \equiv 1 \pmod n$;

5) Inserts (r_i, s_i) $(1 \leqslant i \leqslant v+1)$ in the batch $\mathcal{MS}$.

Such group of messages will be defined as *illegally compensated messages*.

Theorem 2. *If in $\mathcal{MS}$ there is (m, s) with $s^e \neq h(m)$ and m is inserted between r_1 and r_{v+1}, then Algorithm 1 will detect also illegally compensated messages.*

Proof. We shall consider here the set $\mathcal{MS}$ to be totally ordered using an order relation $<$, and — without loss of generality — that

$$(m_1, s_1) < (m_2, s_2) < \cdots < (m_p, s_p).$$

The role of one illegal message (m, s) is to fail Harn's test from the beginning of step 2 in Algorithm 1. This algorithm will continue identifying the first illegal signature (with respect to this order relation) from $\mathcal{MS}$. If in this relation the wrong message (m, s) is positioned between (r_1, s_1) and (r_{v+1}, s_{v+1}), then (r_1, s_1) will be detected and avoided. So, the compensation formula will be destroyed and — beginning with the second run of Algorithm 1, all the illegal signed messages will be detected and identified (because they are no more illegally compensated).

We point out that if (m, s) is positioned after the message (r_{v+1}, s_{v+1}) then it is possible that the illegally compensated messages not to be detected: the case when all these messages are in the first half of the batch, whereas (m, s) is in the second half. In this case Algorithm 1 will detect and avoid (m, s), and all the illegal compensated messages are non-detected.

The LHC scheme can be used for the detection of illegally compensated messages, but — as we have seen — it will be not able to identify them in most of the cases.

Obviously, this type of attack is negligible if h is a one-way hash function (therefore it resists to any PreImage attack).

4.3 Generalization on the Space of Signatures

Algorithm 1 can be easily extended for any homomorphic signature scheme.

Namely, a signature scheme $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is homomorphic if

- $(\mathcal{P}, *)$ and $(\mathcal{A}, \circ)$ are abelian monoids;
- $(\forall K \in \mathcal{K})[sig_K : \mathcal{P} \to \mathcal{A}$ is a morphism];
- $(\forall K \in \mathcal{K})$ $ver_K(sig_K(\alpha) \circ sig_K(\beta)) = ver_K(sig_K(\alpha)) \wedge ver_K(sig_K(\beta))$.

In this case, Algorithm 1 can be rewritten with minor modifications. Of course, it remains to be analyzed — for each case — the problems of complexity and efficiency, compared to the brute force algorithm of separately checking each signature from all messages from the batch.

5 Batch Verifying Scheme for Detecting Illegal Signatures in ECDSA Signature Protocol

The idea to construct batch signature schemes based on pairings appeared in 2006[9,11]. However, a complete scheme for signatures defined on elliptic curves is not definite yet. Taking into account a scheme proposed by Harn for DSA[25], we propose here a batch verifying scheme for a modified form of the Elliptic Curve Signature Standard Algorithm (ECDSA).

A formal definition of ECDSA standard is:

- *Setup.*

– p be a prime or a power of 2, and E be an elliptic curve defined over $\mathbb{Z}_p$.

– P be a point on E having prime order q (such that the Discrete Logarithm Problem in $G = \langle P \rangle$ is infeasible).

– $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$,
$\mathcal{K} = \{(p, q, E, P, a, Q) \mid a \in \mathbb{Z}_{q-1}^*,\ Q = aP\}$.

The values $(p,\ q,\ E, P, Q)$ form the public key, and a is the private key.

- *Signing Algorithm.*

For $K = (p, q, E, P, a, Q)$ and $k \in \mathbb{Z}_q$ is random and secret, define

$$sig_K(m, k) = (\alpha, \beta),$$

where

$$kP = (u, v), \quad \alpha = u \pmod q,$$

$$\beta = k^{-1}(SHA1(m) + a \times \alpha) \pmod q.$$

• *Verification Algorithm.*

For $m \in \{0,1\}^*$, $\alpha, \beta \in \mathbb{Z}_q^*$, the verification of if (α, β) is a signature for the message m is defined by the following computations:

– $w = \beta^{-1} \pmod q$,
$i = w \times SHA1(m) \pmod q$,
$j = w \times \alpha \pmod q$,
– $(u, v) = iP + jQ$,
– $ver_K(m, (\alpha, \beta)) = T \Leftrightarrow u \pmod q = \alpha$.

Any algorithm for signature checking of a batch $\mathcal{MS}$ signed messages cannot ignore the computation of points $iP + jQ$, hence its complexity will not be lower than the one-by-one verification of signatures. The basic problem arises from the type of group defined on the elliptic curve E.

In order to build a batch verifying scheme for detecting illegal signatures, we will perform a small change in the definition of ECDSA standard, by introducing a bilinear pairing which will transfer the additive operation from the group of elliptic points into a multiplicative group.

The modifications of the ECDSA standard are:

• *Setup.* A bilinear pair $e : G \times G \to \mathbb{Z}_q^*$ is added.

• *Signature.* The computation of kP is avoided and we define α as $\alpha = [e(P, Q)]^k$.

• *Verification.*

– Compute $iP + jQ = (u, v)$;
– $ver_K(m, (\alpha, \beta)) = T \Leftrightarrow e((u, v), Q) = \alpha$.

Obviously

$$\alpha = e((u, v), Q) = e(kP, Q) = [e(P, Q)]^k.$$

Algorithm 2.

Input: $\mathcal{MS} = \{(m_i, (\alpha_i, \beta_i)) \mid 1 \leqslant i \leqslant p\}$, $e : G \times G \to \mathbb{Z}_q^*$ a bilinear pairing.

1. $X = \prod_{t=1}^{p} \alpha_t$, $Y = \sum_{t=1}^{p} P_t$ where $P_t = i_t P + j_t Q$;
2. **If** $X = e(Y, Q)$ **then Return** "No error"
else
a) $low \leftarrow 1$, $high \leftarrow p$;
b) $mid \leftarrow \lfloor (low + high)/2 \rfloor$;
c) $X = \prod_{t=low}^{mid} \alpha_t$, $Y = \sum_{t=low}^{mid} (i_t P + j_t Q)$;
d) **If** $X = e(Y, Q)$ **then** $low \leftarrow mid + 1$
else $high \leftarrow mid$;
e) **If** $low < high$ **then goto** 2(b);
f) **Return** $(\alpha_{low}, \beta_{low})$ "is an illegal signature for" m_{low}.
3. **Stop**.

Justification. If k_t is the random parameter generated in the signature phase, and (α_t, β_t) is the signature of message m_t $(1 \leqslant t \leqslant p)$, then

$$\begin{aligned}\sum_{t=1}^{p}(i_t P + j_t Q) &= \sum_{t=1}^{p}(\beta_t^{-1} m_t + \beta_t^{-1}\alpha_t a)P \\ &= \sum_{t=1}^{p} k_t (m_t + a \times \alpha_t)^{-1}(m_t + a \times \alpha_t)P \\ &= \sum_{t=1}^{p} k_t P = \sum_{t=1}^{p} P_t,\end{aligned}$$

and

$$e\Big(\sum_{t=1}^{p} P_t, Q\Big) = \prod_{t=1}^{p} e(P_t, Q) = \prod_{t=1}^{p} [e(P, Q)]^{k_t}.$$

Example 4. Let us consider three messages m_1, m_2, m_3 with signatures $\alpha_s = [e(P, Q)]^{k_s}$,

$$\beta_s = k_s^{-1}[SHA1(m_s) + a\alpha_s], \quad s = 1, 2, 3.$$

We suppose that an intruder replaces the signature of m_3 with (α_3', β_3'). The receiver will verify all signatures using Algorithm 2, as follows:

1) Computes $P_1 = i_1 P + j_1 Q$, $P_2 = i_2 P + j_2 Q$, $P_3' = i_3' P + j_3' Q$;

2) Verifies if $\alpha_1 \times \alpha_2 \times \alpha_3' = e(P_1, Q) \times e(P_2, Q) \times e(P_3', Q)$ holds. The test fails because $\alpha_3' \neq e(P_3', Q)$.

3) Begins step 2 of Algorithm 2:

a) $low \leftarrow 1$, $high \leftarrow 3$, $mid \leftarrow 2$;

b) Verifies if $\alpha_1 \times \alpha_2 = e(P_1, Q) \times e(P_2, Q)$ holds.

c) The answer is "True"; therefore $low \leftarrow 3$ and Algorithm 2 returns "(α_3', β_3') is an illegal signature for m_3".

5.1 Complexity of Algorithm 2

A comparison between the complexity of ECDSA standard and Algorithm 2 is difficult to be established.

• In general the bisection algorithm doubles — for large values of p — the number of computations.

• The modified version of ECDSA replaces the computing of the point kP on the elliptic curve with the computing of a value $e(P, Q)$ (the same for all signatures) and one exponentiation in $\mathbb{Z}_q$. In [29] there are at least 20 schemes for computing multiples of points on elliptic curves, with different complexities, which depend on the type of curve and its parameters.

But generally speaking, our opinion is that Algorithm 2 has the same order of complexity as the application of ECDSA scheme for p times. So, the final decision of which scheme to be chosen for authentication and find illegal signatures will be taken by the users.

6 Conclusions

In this article, the complexity of LHC batch verifying schemes was analysed, and a new batch verifying

scheme for identifying illegal signatures was proposed, in two variants: for RSA signature algorithm — by completing Harn's and LHC's schemes with a bisection algorithm, and for a modified ECDSA standard. Because the idea of verifying in one step the authenticity of several signatures is very appealing, it can be furthermore exploited in different areas of security, for example, for signcryption schemes or for homomorphic vote protocols (if we consider only two directions). It remains for the future to decide the applicability advantages of this modality to reduce the complexity of some verifying authenticity schemes.

Acknowledgments I would like to thank the anonymous referees for their valuable comments and useful remarks about this paper.

References

[1] Bellare M, Garay J A, Rabin T. Fast batch verification for modular exponentiation and digital signatures. In *Lecture Notes in Computer Science 1403*, Nyberg K (ed.), Springer-Verlag, 1998, pp.236-250.

[2] Boyd C, Pavlovski C. Attacking and repairing batch verification schemes. In *Proc. the 6th ASIACRYPT*, Dec. 1976, pp.58-71.

[3] Camenish J, Hohenberger S, Pedersen M. Batch verification of short signatures. In *Proc. the 26th EUROCRYPT*, May 2007, pp.246-263.

[4] Hwang M, Lee C, Tang Y. Two simple batch verifying multiple digital signatures. In *Proc. the 3rd ICICS*, Nov. 2001, pp.233-237.

[5] Lim C H. Efficient multi-exponentation and application to batch verification of digital signatures. 2000, http://dasan.sejong.ac.kr/∼chlim/pub/multi_exp.ps.

[6] Kim K, Yie I, Lim S, Park H. A method of finding bad signatures in an RSA-type batch verification. *Informatica*, 2011, 22(2): 189-201.

[7] Kirn K, Yie I, Lim S, Nyang D. Batch verification and finding invalid signatures in a group signature scheme. *International Journal of Network Security*, 2011, 13(2): 61-70.

[8] Lee S, Cho S, Choi J, Cho Y. Efficient identification of bad signatures in RSA-type batch signature. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, 2006, E89-A(1): 74-80.

[9] Matt B. Identification of multiple invalid signatures in pairing-based batched signatures. In *Proc. the 12th Int. Conf. Practice and Theory in Public Key Cryptography*, March 2009, pp.337-356.

[10] Bao F, Lee C, Hwang M. Cryptanalysis and improvement on batch verifying multiple RSA digital signatures. *Applied Mathematics and Computation*, 2006, 172(2): 1195-1200.

[11] Cao T, Lin D, Xue R. Security analysis of some batch verifying signatures from pairings. *International Journal of Network Security*, 2006, 3(2): 138-143.

[12] Hwang M, Lin I, Hwang K. Cryptanalysis of the batch verifying multiple RSA digital signatures. *Informatica* (*Lithuanian Academy of Sciences*), 2000, 11(1): 15-19.

[13] Lim C, Lee P. Security of interactive DSA batch verification. *Electronics Letters*, 1994, 30(19): 1592-1593.

[14] Selvi S S D, Vivek S S, Shriram J *et al.* Security analysis of aggregate signature and batch verification signature schemes. *IACR Crptology ePrint Archive*, 2009, http://eprint.iacr.org/2009/290.pdf.

[15] Stanek M. Attacking LCCC batch verification of RSA signatures. *IACR Cryptology ePrint Archive*, 2006, http://eprint.iacr.org/2006/111.pdf.

[16] Ferrara A, Green M, Hohenberger S, Pedersen M. On the practicality of short signature batch verification. *IACR Cryptology ePrint Archive*, 2008, http://eprint.iacr.org/2008/015.pdf.

[17] Cheon J H, Kim Y, Yoon H J. A new ID-based signature with batch verification. *Cryptology ePrint Archive*, 2004, http://eprint.iacr.org/2004/131.pdf.

[18] Cui S, Duan P, Chan C W. An efficient identity-based signature scheme with batch verifications. In *Proc. the 1st InfoScale*, May 29-June 1, 2006, Article No.22.

[19] Yoon H, Cheon J H, Kim Y. Batch verifications with ID-based signatures. In *Proc. the 7th ICISC*, Dec. 2004, pp.233-248.

[20] Tzeng S, Lee C, Hwang M. A batch verification for multiple proxy signature. *Parallel Processing Letters*, 2011, 21(1):7 7-84.

[21] Zhang C, Ho P, Tapolcai J. On batch verification with group testing for vehicular communications. *Wireless Network*, 2011, 17(8): 1851-1865.

[22] Changchien S W, Hwang M. A batch verifying and detecting multiple RSA digital signatures. *International Journal of Computational and Numerical Analysis and Applications*, 2002, 2(3): 303-307.

[23] Fiat A. Batch RSA. In *Proc. the 9th CRYPTO*, August 1989, pp.175-185.

[24] Harn L. Batch verifying multiple RSA digital signatures. *Electronics Letters*, 1998, 34(12): 1219-1220.

[25] Harn L. Batch verifying multiple DSA digital signatures. *Electronics Letters*, 1998, 34(9): 870-871.

[26] Li C, Hwang M, Chen S. A batch verifying and detecting the illegal signatures. *International Journal of Innovative Computing, Information and Control*, 2010, 6(12): 5311-5320.

[27] Stinton D. Cryptography Theory and Practice (2nd edition). Chapman & Hall/CRC, 2002.

[28] Cohen H, Frey G. Handbook of Elliptic and Hyperelliptic Curve Cryptography (Discrete Mathematics and Its Applications). Chapman & Hall/CRC, 2005.

[29] Hankerson D, Menezes A, Vanstome S. Guide to Elliptic Curve Criptography. Springer Verlag, 2004.

Adrian Atanasiu is a full professor in computer science at the Bucharest University, Romania since 2000. He received the Ph.D. degree in computer science from the same University in 1978. He was awarded by Romanian Academy with "Moisil Prize" in 2010 for scientific activity. His scientifiec activity includes several stages of research in universities from USA, France, Finland, and Spain. He has published 67 papers and 19 books. His research interests include electronic signatures, elliptic curve cryptography, coding theory, formal languages and applications.