Local and Outsourced Simultaneous Verification of Pairing-based Signatures

Tomasz Hyla

Faculty of Computer Science and Information Technology,
West Pomeranian University of Technology in Szczecin, Poland
thyla@zut.edu.pl

Abstract

Many server-side applications require verifying a large number of digital signatures. In this paper, a practical problem of simultaneous verification of a large number of signatures (up to one hundred thousands) based on pairing-based cryptography is analysed. Based on three exemplary signature schemes, the options for outsourcing computationally intensive operations are presented together with a proposal of verification algorithms that outsource bilinear pairings computation and elliptic curve scalar multiplications. The experimental results from different scenarios of simultaneous verification of a large number of signatures are presented. The test scenarios include verification using a local server, using batch verification, by outsourcing computations to a trusted cloud and by secure outsourcing to possibly dishonest (untrusted) clouds.

Keywords: Secure outsourcing, Bilinear pairing, Signature verification, Cloud, Certificateless signature

1 Introduction

Many services and applications require verifying a large number of a digital signatures. This mainly concerns server-side applications, where it is required to verify that input documents are digitally signed. In case of pairing-based signature' schemes [1-5], time required to create or verify one signature is on acceptable level (on modern computers the time rarely exceed half a second for almost all schemes). The situation for server-side applications is different, sometimes a server must verify thousands of signatures per second. Additionally, a server workload is often uneven, sometimes the number of requests is low and the other time rises sharply. When a server workload is uneven, a server should have high performance, so a server-side application is able to deal with the highest possible number of requests. However, from an economical point of view this approach is highly inefficient. It would be better to outsource the most computational depending operations or to dynamically acquire new resources from a cloud. In such case, the privacy of data [6] and correctness of computations must be ensured.

A cloud can be trusted or untrusted. A trusted cloud does not deviate from its advertised functionality (also it keeps data private). On the other hand, a cloud can be malicious. A malicious cloud is a corrupted cloud that can arbitrary deviate from its advertised functionality (is an active adversary). Additionally, it is possible to specify a semi-honest cloud (also called honest-but-curious) which do not deviate from its advertised property, but it records all the information that should remain private [7] (is a passive adversary). Thus, an untrusted cloud (also called possibly dishonest or potentially malicious) is a cloud that might or might not behave maliciously. Usually this type of cloud reflects public clouds that may lie and misreport about their service quality [8] and there are no guarantees that such a cloud is secure.

The pairing-based signature schemes internally use some operations that might require several millisecond to complete, because of their complexity. The most time-consuming operations are elliptic curve scalar multiplication and bilinear pairing computation. Several solutions can be used to accelerate verification time, especially when a large number of signatures must be verified simultaneously. Firstly, it is possible to outsource entire verification algorithm to a cloud. Secondly, only a computationally intensive operation can be outsourced. However, outsourcing requires having a fully trusted cloud or using secure computation delegation techniques. The secure computation delegation techniques (secure outsourcing) enable an outsourcer to verify results (i.e. an outsourcer can verify if returned result is correct) and to obscure arguments that are sent to a cloud.

1.1 Contribution and Motivation

The motivation for this paper is to analyse methods that can be used to accelerate computation in pairing-based signature schemes and provide conclusions

*Corresponding Author: Tomasz Hyla; E-mail: thyla@zut.edu.pl
DOI: 10.3966/160792642019072004017

based on experimental results. Let's imagine a real scenario when a tax office requires digital signatures based on pairings in every electronic tax form. Taxpayers send different forms during the year to the tax office. However, in every month there is a deadline for sending sales tax settlements. During that period, the number of tax forms received by the tax office per minute increases a hundredfold. The tax office's system uses computation outsourcing to handle increased number of computations (mostly digital signature verifications). In that way, it is not necessary to have additional servers through idle periods. The office is using cryptographic techniques that guarantee information security, because of the sensitive nature of the tax forms. These techniques are described and tested further in this paper. Another scenario that requires fast verification of many signatures involves cryptocurrencies. In cryptocurrencies based on blockchains, digital signatures are used for transactions' confirmation. Mining new coins involves verifications of all transactions that are a part of a current block.

The main contributions are conclusions from several experiments that cover different outsourcing scenarios of pairing-based digital signatures. For the purpose of the experiments, three exemplary digital signature schemes were chosen: CLS scheme [4], CBS scheme II [2], IE-CBHS [5]. In the experiments, the execution time of verification algorithms was tested for various number of input digital signatures ranging from 10 to 100 thousands. The schemes belong to the different categories of pairing-based digital signature schemes and are digital signatures with appendix. The schemes are secure in security models that require resistance against the most advanced adversaries. When implementation of pairing-based signature scheme is required, the schemes would be a first choice for system designers. The identity-based signature schemes [9] were not considered due to their inherent key escrow problem.

The paper discussed different options for outsourcing time-consuming operations and proposes modified versions of verification algorithms (for CLS scheme, CBS scheme II, IE-CBHS scheme). In the modified versions of the algorithms, bilinear pairings and elliptic curve scalar multiplications are outsourced. The outsourcing methods to untrusted (dishonest) clouds have secrecy and verifiability properties, so the cloud is not able to get any information from a computation request and results can be verified by a local server.

The proposed algorithms were implemented and tested in several experiments. The experiments include verification using a local server, batch verification, generic outsourcing, by outsourcing computations to a trusted cloud and by secure outsourcing of computations to possibly dishonest clouds.

1.2 Paper Structure

The remainder of this paper is organized as follows. Section 1.3 contains information about related work concerning server-aided signatures, batch verification, outsourcing of bilinear pairings and elliptic curve scalar multiplications. Section 2 contains description of verification algorithms from the three schemes and describes different solutions that can accelerate verification speed. Section 2 ends with discussion about security of proposed solutions. Section 3 contains experimental results for algorithms described in Section 2. The test were done on laboratory computers that acted as a cloud and were extrapolated to a high performance cloud. The paper ends with conclusion about different solutions to the problem of simultaneous verification of large pairing-based digital signatures' sets.

1.3 Related Work

1.3.1 Server-aided Verification

Girault and Lefranc [10] introduced a Server-Aided Verification (SAV) concept, which allows delegating a substantial part of computations to an untrusted powerful server (cloud). Informally, an untrusted server is a server (or a cloud) under control of a third party that provides computing services. The third party can declare or not compliance to some security standards. Generally, there are no guarantees that the server will return correct results. Hence, in applications that require high security levels, additional security measures must be applied. SAV signature scheme consists of a digital signature scheme and a server-aided verification protocol. Wu et al. [11-12] defined more advanced models for the security of server-aided verification signatures and introduce existential unforgeability of server-aided verification signatures (EUF-SAV-Σ). Also, they proposed SAV schemes for Waters [13] and BLS [14] signatures. Chow et al. [15] proposed a new model to capture the collusion attack and provided a generic construction of SAV applicable on a wide class of pairing-based cryptosystems. For example, Qin et al. [16] used server-aided verification technique to speed up payment verification in a mobile wallet.

1.3.2 Outsourcing of Bilinear Pairing Computation

The secure pairing delegation algorithms designed for devices with limited capability was presented by Chevallier-Mames et al. [17]. The algorithm achieves unconditional security, but it requires calculating several scalar multiplications and exponentiations during preparation and verification phase. The total computation time of those operations is longer than pairing calculation. The main advantage of this algorithm is lack of necessity to implement pairing operation. Conard et al. [18] proposed efficient

versions of a pairing delegation algorithm which have secrecy and verifiability properties. However, the best estimated efficiency ratio when both pairing arguments A and B are variable is 0.92 and 0.30 when one of them is constant and public.

Chen et al. [19] presented algorithm *Pair*, which does not require computationally expensive operations. Algorithm *Pair* is secure in the one-malicious version of two untrusted program (OMTUP) model [20]. In this model computations must be split into two parts, which are sent to two different servers U_1 and U_2. One of the servers must be honest for the algorithm to be secure. Moreover, some precomputation using subroutine *Rand* is required. Tian et al. [21] proposed two improved algorithms comparing to *Pair*. One of the algorithm is more efficient (*Algorithm A*) and second one has improved verifiability of results (*Algorithm B*). Similarly, the algorithms require precomputation using *RandA* and *RandB* subroutines. In 2017, Dong et al. [22] proposed algorithm *DBP* that is fully verifiable, but is less efficient that *Pair* and *Algorithm A* [21]. Recently, in 2018 Dong and Ren [23] proposed algorithm *BPS* that uses only single untrusted cloud server with checkability (verifiability) close to one, but with higher computational costs.

In some cases, it is not necessary for a secure outsourcing scheme to have verifiability property, e.g., in encryption schemes where correctness of pairing verification can be computed later using other means. Guillevic and Vergnaud [24] proposed two efficient protocols for secret pairing delegation without verifiability. In 2016 Luo et al. [25] proposed a generic scheme for securely outsourcing multi-bilinear pairings.

1.3.3 Outsourcing of Elliptic Curve Scalar Multiplication

Hohenberger and Lysyanskaya [20] proposed secure outsourcing schemes for scalar multiplication in the one-malicious version of two untrusted program model. More efficient scheme *Exp* in that security model with higher verifiability was proposed by Chen et al. [26] [27]. Wang et al. [28] proposed an efficient scheme for securely outsourcing modular exponentiations in single untrusted program model, but it is difficult to translate this scheme into an elliptic curve scalar multiplication problem (in some of the other schemes it is a trivial task). In 2017, Ding et al. [29] proposed *MExp* algorithm for secure outsourcing of simultaneous modular exponentiations. Other works related to modular exponentiation include [30-31].

In 2016, Zhou and Ren proposed *SecMul* scheme [32] for secure outsourcing of elliptic curve scalar multiplication. The *SecMul* is highly efficient, does not require precomputation and its security is based on the hardness of integer factorization problem. However, it does not provide possibility to verify results returned from the cloud and it assumes that p, one of an elliptic curve parameters, is secret (p is an integer which specify the finite field F_p on which elliptic curve $E(F_p)$ is built). In most of cryptographic applications, elliptic curves with standardized and publicly known parameters are used, e.g. [33], so *SecMul* cannot be used in such cases.

1.3.4 Batch Verification

A batch verification algorithm verifies a set of (message, signature) pairs as a group [34-38]. A batch verification can reduce the number of computationally intensive operations, but provides one result for all signatures. If one of the signatures is invalid, the procedure must be restarted using one of the strategies [34] (e.g., using binary splitting, where basically a signature set is divided into two parts and the algorithm is restarted for each part). The first batch verification algorithm was proposed by Fiat [39], followed by works of Harn [40]. Batch verification for ID-based signatures was proposed by Yoon et al. [41] and Shi et al. [42]. Batch verification algorithms for certificateless schemes was proposed by Geng and Zhang [43] and Fan et al. [44].

2 Different Approaches to Simultaneous Verification of Pairing-based Signatures

Verification algorithms for different schemes built using pairings contain computationally expensive operations, e.g., computation of a bilinear pairing, computation of elliptic curve scalar multiplication and exponentiations in multiplicative group. In many cases, it is possible to do scalar multiplications instead of exponentiation in multiplicative group G_T using properties of bilinear maps. Other operation like modular addition and multiplication in Z_{p^*}, addition in G_1, multiplication in G_T and hashing are several orders of magnitude faster.

Acceleration of verification speed can be achieved in several ways. The easiest and the most obvious way is to use more powerful servers, but if high processing power is not needed permanently, this approach is economically inefficient. The second option is to use parallel processing. The third option is to use batch processing that provides one result for a set of signatures. The next option is to outsource the computation of verification algorithm or only the most time consuming operations to a trusted cloud. If only dishonest (untrusted) clouds are available, the computationally intensive operations can be outsourced using secure outsourcing techniques.

Three pairing-based signature schemes were chosen for the purpose of this paper to test different options of accelerating the verification algorithms, when a large number of signatures must be verified in a short time. The definitions and notations related to bilinear

pairings can be found in [1]. The following schemes were analysed:

- a certificateless scheme: CLS scheme [4] (**CLS**) – secure against Super A_I and Super A_{II} adversaries in the random oracle model;
- an implicit certificate based scheme: CBS scheme II [2] (**CBS**) - secure in the random oracle model against Super-CB-A_I and Super-CB-A_{II} adaptive chosen message and chosen identity attacks;
- an implicit and explicit certificate based: IE-CBHS [5] (**IE-CBHS**) - existentially unforgeable against adaptive CMA in the random oracle model with Super Type I and Type II adversaries.

Also, Huang et al. [3] proposed a certificateless scheme with the same security level as Zhang et al. [4]. However, a verification algorithm in that scheme is mathematically identical to the algorithm in CBS scheme [2], so the results will be identical to that from CBS scheme.

2.1 Verification Algorithms for Chosen Signature Schemes

The CBS scheme verification algorithm CB-Verify is as follows [2]:

Input: *params*, a message/signature pair (m, $\sigma = (u, v, W)$), ID's public key PK_{ID}

Compute: $\widehat{ID} = H_2(ID, PK_{ID})$

Return true if:

$$u \equiv H_1(m, \widehat{ID}, PK_{ID}, vp + uPK_{ID}, \hat{e}(W,P)\hat{e}(mpk, H_0(\widehat{ID}))^u) \quad (1)$$

Notation: H_0, H_1, H_2 – secure cryptographic hash functions, $\hat{e}$ – bilinear pairing, mpk – master public key, P - a generator of G_1^*.

The CLS scheme verification algorithm CLS-Verify is as follows [4]:

Input: *params*, identity ID_i public key P_i, message m_i, signature $\sigma = (R, V)$

Compute:

$$Q_i = H_1(ID_i, P) \quad (2)$$

$$u = H_2(R, P_i, m_i) \quad (3)$$

$$v = H_3(R, P_i, m_i) \quad (4)$$

Return true if:

$$\hat{e}(V, P) \equiv \hat{e}(uP_i + vP_T + R, Q_i) \quad (5)$$

Notation: H_1, H_2, H_3 – secure cryptographic hash functions, $\hat{e}$ – bilinear pairing, P_T – master public key, P - a generator of G_1^*.

The IE-CBHS scheme verification algorithm Verify is as follows [5]:

Input: *params*, A_{ID}, R_{ID}, Pk_{ID}, certificate information CI_{ID}, message/signature/certificate triple (m, $\sigma = (h, w, W)$, $cert_{ID}$)

Compute:

$$q_{ID} = H_1(CI_{ID}, Pk_{ID}, A_{ID}, R_{ID}) \quad (6)$$

$$Q_{ID} = H_2(CI_{ID}, Pk_{ID}, A_{ID}, R_{ID}, q_{ID}) \quad (7)$$

$$U' = \hat{e}(W,P)\hat{e}(Q_{id}, q_{ID}^{-2}(cert_{ID}P - A) + (1 - q_{ID}^{-1})R_{ID})^h \quad (8)$$

$$\overline{k_1 P} = wP + hPK_{ID} (\text{mod } p) \quad (9)$$

Return true if:

$$h \equiv H_3(m, \overline{k_1 P}, U', Q_{ID}) \quad (10)$$

$$A_{ID} \equiv cert_{ID}P - (R_{ID} + q_{ID}P_0)q_{ID} \quad (11)$$

Notation: H_1, H_2, H_3 – secure cryptographic hash functions, $\hat{e}$ – bilinear pairing, P_0 – master public key, P - a generator of G_1^*, Pk_{ID} – a public key of a user with identity ID, A_{ID}, R_{ID} – public user parameters.

2.2 Batch Verification

The batch verification of signatures allows verifying n signatures simultaneously. The batch verifier returns true if all signatures are valid and false if one or more is invalid (for formal definition of Batch Verification of Signatures see Qin et al. [45]). According to Yoon et al. [41], input of batch verification can be classified into three types: (1) multiple signatures on a single message generated by multiply signers, (2) multiple signatures on multiple messages generated by a single signer, (3) multiple signatures on multiple messages generated by multiple signers, where a distinct user signs each message.

Informally, security models for these three types of input to batch verifiers capture scenarios where adversaries are able to create two or more invalid signatures that mutually cancel themselves out when they are added or multiplied. If adversaries are able to produce such signatures, then the result of batch verification will be positive, instead of being negative.

The batch verification algorithm for CLS scheme [4] was proposed by Geng and Zhang [43]. They use technique introduced by Qin et al. [45] that uses additional random exponents for batch pairing verification. The scheme is slightly modified to support batch verification of Type 3 (the strongest security requirement) and is proven existentially unforgeable against adaptive chosen-message attacks under the standard computational Diffie-Hellman assumption.

Other tested schemes (CBS, IE-CBHS) do not have batch verification algorithms. This mainly results from

the fact, that equations in the verification algorithms are not linear, i.e. it is not possible to add or multiply both sides of equations.

The algorithm **CLS-Batch-Verify** [43] is as follows: **Input:** *params*, *n* tuples < public key P_i, message m_i, signature σ_i = (R_i, V_i) > Randomly choose a vector $\delta = (\delta_1, \delta_2, ..., \delta_n)$ with each $\delta_i \in \{0,1\}^l$ from Z_q^*, l is a size of small exponent number;

(1) Compute: $Q_i = H_1(ID_i, P) = H_4(\Delta)$, $u_i = H_2(R_i, P_i, m_i)$, $v_i = H_3(R_i, P_i, m_i)$;

(2) Verify whether the equation holds:

$$\hat{e}(\sum_{i=1}^{n} \delta_i V_i, P) \equiv \equiv \hat{e}(\sum_{i=1}^{n} \delta_i (u_i, P_i + R_i), W) \hat{e}(\sum_{i=1}^{n} \delta_i v_i Q_i P_0) \quad (12)$$

Notation: H_1, H_2, H_3, H_4 – secure cryptographic hash functions, $\hat{e}$ – bilinear pairing, P_0 – master public key, P - a generator of G_1^*, Δ – any public parameter which is the same for every signer.

2.3 Outsourced Versions

The signature verification algorithm might be outsourced using two strategies. In the first one, an entire verification algorithm is outsourced (also called generic outsourcing). In the second one, only the most time-consuming operations are outsourced (computation outsourcing).

Computation outsourcing allows accelerating computation with usage of external servers (or a cloud). The client *T* sends data to server (cloud) *U*, which performs computations and sends back results to *T*. The security of computations depends on the status of *U*, which can be trusted, untrusted (possibly dishonest) or semi-honest. Formal security definitions and models for secure outsourcing of cryptographic computation are described by Hohenberger and Lysyanskaya [20] and Lei et al. [7].

Secure computation outsourcing techniques requires that *T* prepares (transforms) the data before sending it to *U* and then after receiving result *T* verifies it. Hence, *U* does not have possibility to recover original values and *T* can verify if results are correct.

The following procedure was used for preparation of outsourced versions of the verification algorithms. Firstly, the most time-consuming operations were identified, i.e. BP (Bilinear Pairing) and SM (elliptic curve Scalar Multiplication). Secondly, the computations were divided into phases (in a such way that one *T phase* contains fast computations, the next phase is outsourcing time-consuming operations from *T* to *U*; if an input from BP or SM depends on output of another BP or SM then more phases are necessary,). The signatures are verified phase by phase (i.e., *T phase 1* is executed for all signatures, then *U phase 1* is executed for all signatures, then *T phase 2* is executed, etc.) instead of sequentially verifying signatures. This enables to group computation outsourcing' request to *U* and send them as one request.

The following versions of algorithms for *n* signatures simultaneous verification using outsourcing (**nO** – ***n*** signatures **O**utsourcing) were created:

- **nO-CB-Verify** for **CBS** scheme (Table 1);
- **nO-CLS-Verify** for **CLS** scheme (Table 2);
- **nO- IE-CBHS-Verify** for **IE-CBHS** scheme (Table 3).

The above algorithms are basically verifying *n* signatures at once and are divided into phases with extracted intensive operations. However, the algorithms work the same as their original versions. Hence, if they are executed in trusted environment (using secure outsourcing or trusted cloud), their security model will not change.

Table 1. nO-CB-Verify algorithm

nO-CB-Verify	
Input	*params*, *n* tuples (a message m_i, a signature $\sigma_i = (u_i, v_i, W_i)$, a public key PK_i)
T phase 1	$\widehat{ID}_l = H_2(ID_i \parallel PK_i)$, where *i=1...n*
U phase 1	$sm_{i,1} = SM(P, v_i)$ $sm_{i,2} = SM(PK_i, u_i)$ $sm_{i,3} = SM(\widehat{ID}_i, u_i)$ where *i=1...n*
T phase 2	$t_i = sm_{i,1} + sm_{i,2}$, where *i=1...n*
U phase 2	$e_{i,1} = BP(W_i, P)$ $e_{i,2} = BP(mpk_i, sm_{i,3})$ where *i=1...n*
T phase 3	$t_{i,2} = e_{i,1} e_{i,2}$, where *i=1...n* Return true if: $H_1(M_i, \widehat{ID}_l, PK_i, t_{i,1}, t_{i,2})$, where *i=1...n*

Table 2. nO-CLS-Verify algorithm

nO-CLS-Verify	
Input	*params*, *n* tuples <identity ID_i, a public key P_i, a message m_i, a signature $\sigma_i = (R_i, V_i)$
T phase 1	$Q_i = H_1(ID_i, P)$ $u_i = H_2(R, P_i, m_i)$ $v_i = H_3(R, P_i, m_i)$ where *i=1...n*
U phase 1	$sm_{i,1} = SM(P_i, u_i)$ $sm_{i,2} = SM(P_T, v_i)$ where *i=1...n*
T phase 2	$t_i = sm_{i,1} + sm_{i,2} + R$
U phase 2	$e_{i,1} = BP(V_i, P)$ $e_{i,2} = BP(t_i, Q_i)$ where *i=1...n*
T phase 3	Return true if: $e_{i,1} \equiv e_{i,2}$, where *i=1...n*

Table 3. nO- IE-CBHS-Verify algorithm

nO- IE-CBHS-Verify	
Input	*params*, *n* tuples $<A_i, R_i, PK_i$, certificate information CI_i, message m_i, signature $\sigma_i = (h_i, w_i, W_i)$, certificate $cert_i>$
T phase 1	$q_i=H_2(CI_i, PK_i, A_i, R_i)$ $Q_i=H_2(CI_i, PK_i, A_i, R_i, q_i)$ $x_{i,1} = q_i^{-1}$ $x_{i,2} = q_i^{-2}$ $x_{i,3} = 1 - q_i^{-1}$ where *i=1...n*
U phase 1	$sm_{i,1} = SM(P, cert_i)$ $sm_{i,2} = SM(P, w_i)$ $sm_{i,3} = SM(PK, h_i)$ $sm_{i,4} = SM(P_0, q_i)$ $sm_{i,5} = SM(Q_i, h_i)$ $sm_{i,6} = SM(R_i, x_{i,3})$ where *i=1...n*
T phase 2	$t_{i,1} = sm_{i,2} + sm_{i,3}$ $t_{i,2} = R_i + sm_{i,4}$ $t_{i,3} = sm_{i,1} - A_i$ where *i=1...n*
U phase 2	$sm_{i,7} = SM(t_{i,2}, q_i)$ $sm_{i,8} = SM(t_{i,3}, x_{i,2})$ where *i=1...n*
T phase 3	$t_{i,4} = sm_{i,8} + sm_{i,6}$, where *i=1...n*
U phase 3	$e_{i,1} = BM(W_i, P)$ $e_{i,2} = BM(sm_{i,5}, t_{i,4})$ where *i=1...n*
T phase 4	Return true if: $h_1 = H_3(CI_i, m_i, t_{i,1}, e_{i,1}, e_{i,2}, Q_i)$ $A_1 = sm_{i,1}, sm_{i,7}$ where *i=1...n*

2.4 Security Discussion

The large number of signatures can be simultaneously verified using several methods (Table 4). Locally, a set of signatures can be verified by sequentially executing *n* time's verification algorithm or batch verification algorithms. The main drawback of a batch verifier is lack of resistance to a denial of signature verification attack, i.e. when an adversary prepares some number of false signatures, a verifier will have to use a standard verification algorithm and verify signatures one by one to find those who are correct.

The generic outsourcing (computing complete verification algorithm on a remote cloud) requires a fully trusted cloud environment. The intermediate approach is to outsource only computationally intensive operations. Such approach requires a trusted cloud, but implementation in a cloud is simpler. If trusted clouds are not available, the only option is to use secure outsourcing techniques.

2.4.1 Using Secure Outsourcing of Computations.

The CLS, CBS, IE-CBHS schemes are secure (according to their security models). The security of algorithms: nO-CB-Verify, nO-CLS-Verify, nO-IE-CBHS-Verify, that outsource BP and SM computation from *T*, depend on outsource *U* status and on type of used outsourcing algorithm. If *U* is a cloud that is trusted (e.g., *U* is a private cloud that is under control of *T*), then computations on *U* have the same security level as computations on *T*. Hence, in case of a trusted cloud the security properties of the schemes do not change.

Table 4. Simultaneous signature verification methods

Method	Computations		Total implementation complexity	Cloud implementation requirements	Cloud security requirements
	local	remote			
local verification	Yes	No	low	N/A	N/A
batch verification	Yes	No	medium	N/A	N/A
generic outsourcing	No	Yes	low	complete verification algorithm	high
outsourcing of computationally intensive operations	Yes	Yes	medium	only some computations: BP, SM	high
secure outsourcing of computationally intensive operations	Yes	Yes	high	only some computations: BP, SM	low
secure outsourcing of computationally intensive operations using *One-malicious version of two untrusted program model*	Yes	Yes	high	only some computations: BP, SM	medium

However, if possibly dishonest (untrusted) public clouds are used, a special algorithm for secure outsourcing must be used. If secure outsourcing algorithm is used, i.e. the algorithm is not only efficient, but also it has a secrecy and verifiability property. Because of that, it is assumed that the outsourced computation has the same security level as computation done on local server *T*.

It is difficult to build such an algorithm for BP or SM that also accelerate computation by a few orders of magnitude. The main difficulty lies in the fact that transformation used by *T* to obscure arguments of BP

or SM before sending them to U are also computationally intensive. Because of that, the one-malicious version of two untrusted program model is currently the only option for practical applications. Informally, this model assumes that one of two servers (clouds) U_1 and U_2 can be dishonest and U_1 and U_2 cannot communicate with each other.

2.4.2 One-malicious Version of Two Untrusted Program model in Practical Applications

In the OMTUP model, it is necessary to add another security assumptions to the schemes, so they will be considered secure, i.e. the implicit assumption that computed locally BP and SM operations are secure must be replaced with the explicit assumption that BP and SM operations are assumed to be secure, when they are calculated using secure outsourcing algorithm in OMTUP model. In case of digital signatures' verification, using the OMTUP model might be acceptable in practical applications. There are a few reasons for that. Firstly, it is possible to choose two different cloud providers (who claim high security standards). Secondly, when only SM and BP are outsourced (like in nO-CB-Verify, nO-CL-Verify, nO-IE-CBHS Verify) and two cloud providers are dishonest and cooperate (so security is compromised), then they will have possibility to know only whose signature is verified (e.g., by comparing the public key values), but not the messages itself. Moreover, they will have possibility to manipulate verification results.

It is not an easy task to manipulate verification results, because outsourcing 100k BP requires calculating 600k pairings (using *AlgorithmA* [21]) on the U_1 an U_2. The BP computations requests are sent using random order and it would be difficult to match BP request from one signature (e.g., in nO-CLS-Verify). Because of nO-CB-Verify and nO-IE-CBHS-Verify algorithms construction, dishonest U_1 and U_2 do not have enough data to generate BP results that could cause T to verify positively a false signature. Of course, dishonest U_1 an U_2 can send false results, that would cause T to negatively verify a correct signature. In such a case, T can restart the verification algorithm locally for each negative result, when a small number of negative verification is expected in the system.

3 Experimental Results

The time required to verify one hundred thousand signatures was measured for three different pairing-based schemes described in Section 2 (CBS [2], CLS [4], IE-CBHS [5]). The tests cover: verification using 1 core and 4 core processing; batch verification (for CLS scheme); generic outsourcing; outsourcing of BP and SM computations to trusted servers (clouds) and outsourcing of BP and SM computations using secure outsourcing using one-malicious version of two untrusted program model.

The test environment consists of a local server (Intel Xeon W3520 2,66GHz, 12 GB RAM). The local server is connected through 1Gbit/s connection to a private cloud. The cloud consists of the server (two Intel Xeon E5504 2GHz processors, 24GB RAM) responsible for providing computing services, which is internally connected to 12 computers with four core processors (Intel Xeon W3520 2,66GHz, 12 GB RAM), which are responsible for computing requested operations by the server (further in this section this cloud is called **Lab Cloud**). The tests were also simulated using the **Simulated Cloud**, which is a simulation of the Lab Cloud with 120 four core processors.

The schemes were implemented using MIRACL library [46]. Type 1 symmetric pairing built on GF(p) curve was used (MR_PAIRING_SSP, AES-128 security GF(p) in MIRACL library). Time was measured using C++ chrono library. All results are the average of three repetitions. The test results include time required for data transfers. In the tests, a text file (10KB) was used as an input message.

3.1 Local Verification

The purpose of the first test was to measure the signature verification time of CBS, CLS and IE-CBHS schemes using only the local server. The Figure 1 presents results for a single thread. The results are presented using a logarithmic scale. The signature verification time for all three schemes is below 0.4s. The time required to verify 100k signatures is 6.6h (23714s) for CBS, 6.0h (21667s) for CLS and 10.8h (38865s) for IE-CBHS.

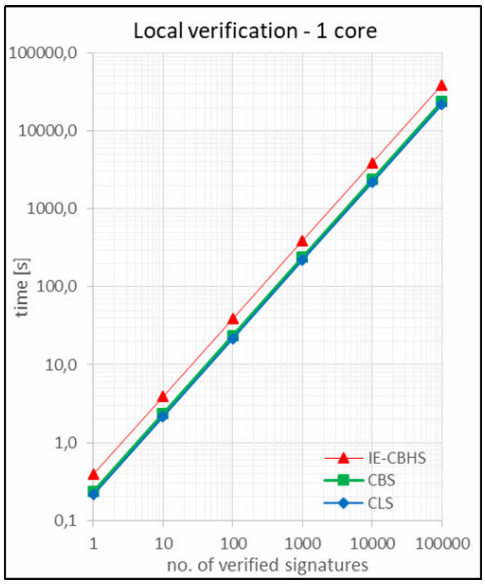

Figure 1. Time required for local signature verification using one CPU core

The Figure 2 presents results for four threads using four physical processor cores. The signature verification is an independent operation, which does

not relay on the results of other signatures' verifications. Because of that, it is easy to create a parallel verification algorithm with almost linear acceleration (using OpenMP). The time required to verify 100k signatures is 1.7h (6081s) for CBS, 1.5h (5556 s) for CLS and 2.8h (9965 s) for IE-CBHS. The results for 1 and 10 signature verifications do not have such acceleration, because each signature verification is run using one thread. Of course, it is possible to accelerate verification of one signature by parallelisation of operations inside each scheme verification algorithms.

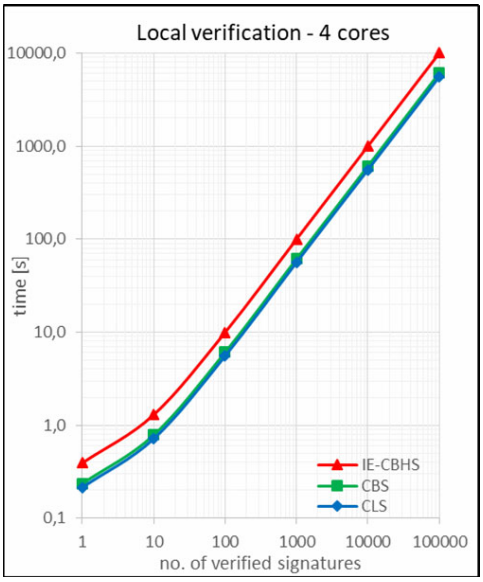

Figure 2. Time required for local signature verification using four CPU cores

3.2 Batch Verification

The batch verification can lower the number of time-consuming operations. However, the major drawback is that the verification result is a binary 0 or 1 result for all signatures together. The Figure 3 presents results (in a logarithmic scale) of batch verification vs standard verification (i.e. executing n times verification algorithm) for CLS scheme (using a single thread). The batch verification algorithm for CLS scheme was 3 times faster (2h (7303s) for 100k signatures' verification instead of 6h (21667s)). It is less than one might expect, but for each signature verification it is still required to calculate two scalar multiplications (SM) – standard verification algorithm must calculate 2BP and 2SM for each signature. Moreover, this was calculated with the assumption that the verification of all signatures is positive. In case of one or more errors, the need to recalculate results, using one of a few strategies (e.g., binary splitting), cause that the advantage of batch verification diminishes fast.

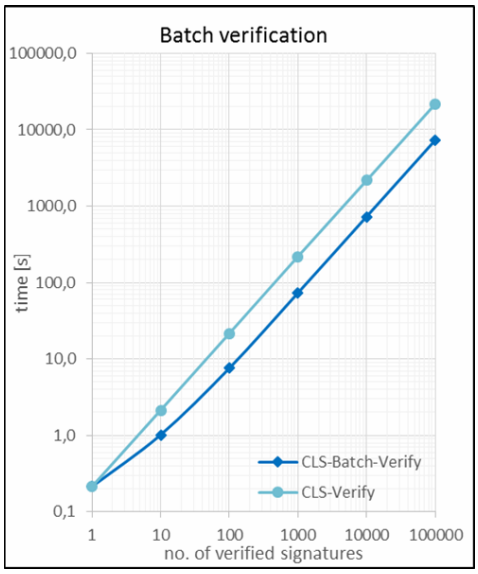

Figure 3. Time required for signature verification using batch verification

3.3 Generic Outsourcing

In a generic outsourcing scenario, the signed files are sent to a cloud which executes verification algorithm for each signed file and returns a result. The test were done using Lab Cloud and 10KB and 1MB files. It is necessary to send signed files to a cloud (hashes are not enough), because hash functions in the schemes use the signed file as one of its input arguments.

The data transfer time for 100k signed 10KB files was below 10s. However, for 1MB file size, transfer time is obviously 100 times longer. Total verification time for 100k signatures is from 7 to 13 min (456-819 s) for 10KB file and from 21 to 26 min (1248-1611 s) for 1MB file (Figure 4 and Figure 5). This method allows fast verification using a trusted cloud, but depends on the signed file size, i.e. in case of large signed files, most of the total computation time will be used for data transfer.

The schemes could be modified (Sign and Verify algorithms) by replacing a message with a hash of the message to avoid long file transfers to a cloud. In such cases a local server must have ability to calculate hashes. Such modification probably will not change the security of the schemes, but would require some security analysis.

3.4 Outsourcing Verification to a Trusted Cloud

In this test scenario scalar multiplication (SM) and bilinear mapping (BP) operations were outsourced to the trusted cloud. The total time required for processing of n signatures consists of time required for local processing t_{local} (operations other than SM and BP), time required for data transfer t_{tr} (in case of 200k simultaneous BP computations request, request sent to a cloud has around 111MB and can be transmitted

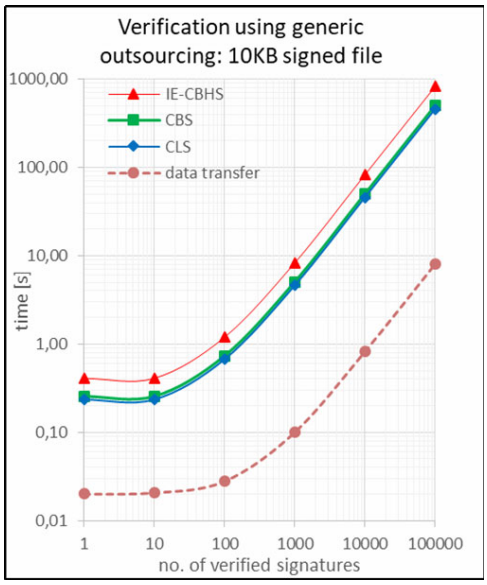

Figure 4. Time required for signature verification using generic outsourcing for 10KB signed file

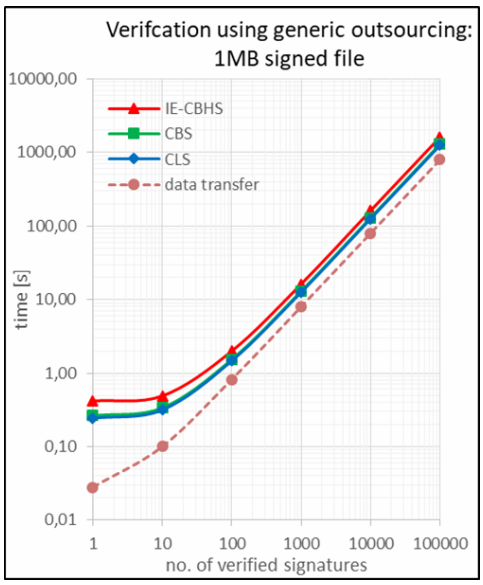

Figure 5. Time required for signature verification using generic outsourcing for 1MB signed file

using 1Gbit/s link in around 1s) and time required for cloud computation t_{cloud}.

The Figure 6 contains results for the Cloud and Figure **7** for the Simulated Cloud (in a logarithmic scale). The Figure 6 and Figure 7 contain also information about non-cloud time ($t_{local} + t_{tr}$). The time required for verification of 100k signatures for CBS scheme is 13 min (787 s), for IE-CBHS is 27 min (1593 s) and for CLS is 10 min (620 s) using the Lab Cloud. The results for the Simulated Cloud are: 4 min (237 s) for CBS, 12 min (734 s) for IE-CBHS, 4 min (256 s) for CLS. They are close to non-cloud time. This means that further increasing a number of CPU cores in the cloud will not decrease total time significantly.

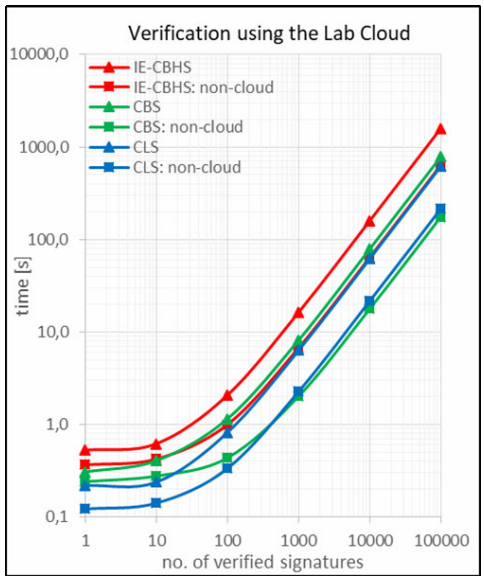

Figure 6. Time required for signature verification using outsourcing to Lab Cloud

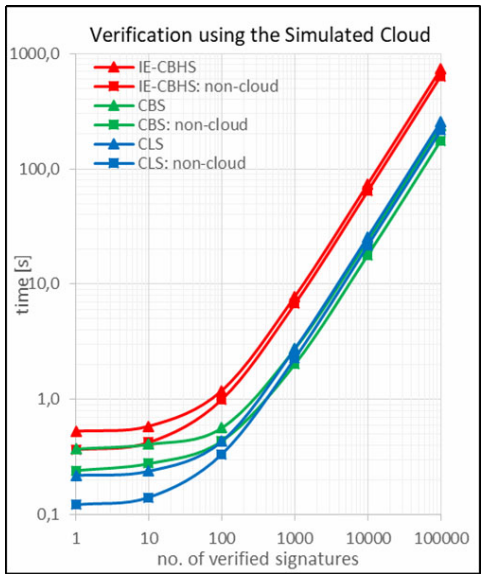

Figure 7. Time required for signature verification using outsourcing to Simulated Cloud

3.5 Verification Using Secure Outsourcing

In the secure outsourcing test scenario verification algorithms were tested using *AlgorithmA* [21] for secure outsourcing of BP operation and algorithm *Exp* [26-27] for outsourcing of SM operation. Both algorithms work in one-malicious version of two untrusted program model. The total time required to process n signatures consists of: local processing t_{local}, time required for problem transformation for BP and SM t_{prep}, time required for data transfer t_{tr}, time required for cloud computation t_{cloud}, time required for verification of cloud computation t_{verify}. Moreover, the *AlgorithmA* generates 6 BP request to clouds U_1 and U_2 and *Exp* generates 6 SM requests. Because of that, a cloud must calculate a few times more operations than in outsourcing to a trusted cloud.

The Figure 8 contains results for the Lab Cloud and Figure 9 for the Simulated Cloud (in a logarithmic scale). The time required for verification of 100k signatures for CBS scheme is 38 min (2291 s), for IE-CBHS is 54 min (3279 s) and for CLS is 33 min (2015 s) using the Lab Cloud. The results for the Simulated Cloud are 10 min (618 s) for CBS, 22 min (1306 s) for IE-CBHS and 9 min (522 s) for CLS. It is not possible to decrease much more the total time using more powerful cloud than the Simulated Cloud, because the non-cloud time (t_{local} + t_{prep} + t_{tr} + t_{verify}) part of operations takes most of the time.

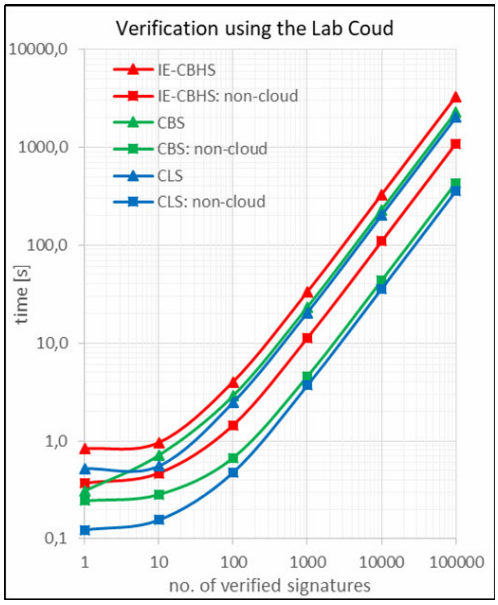

Figure 8. Time required for signature verification using secure outsourcing to Lab Cloud

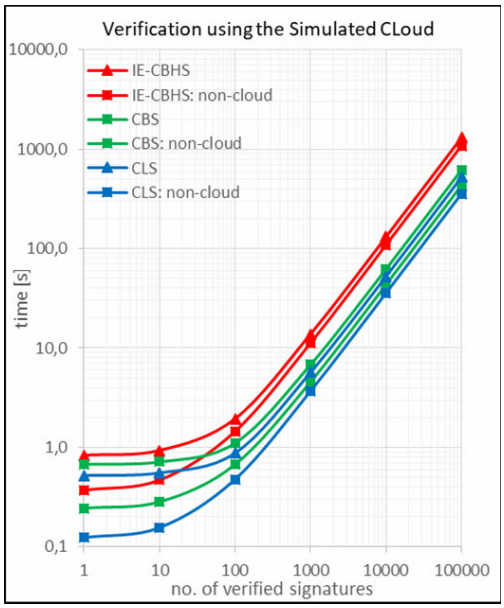

Figure 9. Time required for signature verification using secure outsourcing to Simulated Cloud

Algorithms *AlgorithmA* and *Exp* require precomputations that can be done using offline mode. The time required for precomputations on the local server for *AlgorithmA* using *RandA* generator is 86ms and for *Exp* using *BPV* generator - 5.7ms.

3.6 Comparisons

The acceleration ratio between verification with one CPU core on the local server and verification using secure outsourcing of BP and SM operation is around 10 for Lab Cloud (Table 5) and from 24.4 to 41.5 for Simulated Cloud (Table 6). This ratio is even higher for outsourcing BM and SM computation to a trusted cloud – between 29.8 and 35.0 for Lab Cloud (Table 5) and between 52.9 and 99.8 for Simulated Cloud (Table 6). This is caused by the fact that local server does not need to prepare and verify computation requests that are sent to the cloud.

Table 5. Acceleration ratio for Lab Cloud (100k signatures verification)

from one core server computation to computation using:	CBS	CLS	IE-CBHS
secure outsourcing	10.4	10.8	11.9
outsourcing to a trusted cloud	30.1	35.0	29.8

Table 6. Acceleration ratio for Simulated Cloud (100k signatures verification)

from one core server computation to computation using:	CBS	CLS	IE-CBHS
secure outsourcing	38.4	41.5	24.4
outsourcing to a trusted cloud	99.8	84.7	52.9

Due to the operations that must be computed locally (mainly because of the preparation and verification of computation request sent to cloud), it is not possible to reduce total time for verification of 100k signatures much more than using Simulated Cloud. Only about 20% of total time (Figure 10) was spent on computations in the cloud using secure outsourcing to Simulated Cloud. The 80% of the time was spent on local computations and data transfers. The local computation speed, using secure outsourcing for each BP or SM request, does not depend on a cloud computation power. Other computation in tested schemes and data transfer time do not depend on the cloud as well. This is the reason that using secure outsourcing it is not possible to reduce the total time below a threshold (non-cloud time).

4 Conclusion

The simultaneous verification of a large number of signatures requires a lot of processing power. The total time could be decreased tenfold by using more powerful processors with ten or more processor cores. The same acceleration was achieved in test environment using secure outsourcing of BP and SM operations, which can accelerate computation around

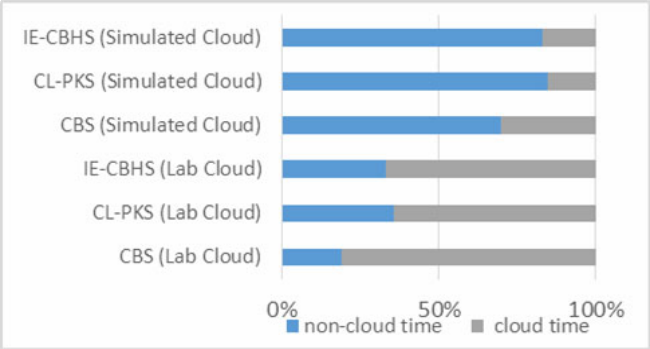

Figure 10. Division of total computation time in secure outsourcing

10 to 40 times depending on the signature scheme and cloud configuration. Using the same cloud configuration, it is possible to achieve 100 acceleration ratio when the local server is 2-3 times more powerful.

The batch verification is an easy method to implement solution for verification of many signatures at once in cases where all verified signatures are expected to be correct. In tested cases, it allows to calculate a scalar multiplication instead of a bilinear pairing for each signature and to reduce the total computation time three times. However, in practice to create a batch verification algorithm, a signature verification algorithm must have a linear structure of an equitation that contains a bilinear pairing or a scalar multiplication computations.

Outsourcing only the most time-intensive operations simplifies the creation process of outsourced versions of verification algorithms. In comparison to SAV protocols designed especially for each scheme, the security of outsourced scheme will depend on the security model used for computations outsourcing. When only some operations are outsourced to a trusted cloud, then in case of some signature schemes like CBS [2] and IE-CBHS [5], the final verification is done locally. Schemes' constructions make it practically impossible for a dishonest cloud to send false results that will cause positive verification of the signature. However, errors from the cloud will cause negative verification results.

Even with outsourcing to a trusted cloud, it will be still difficult to reduce total time for verification of 100k signatures to a few seconds, because of computations that must be done locally and data transfer times. Even if entire verification algorithms is outsourced to a cloud, it will be difficult to decrease total verification time to a few seconds when signed files are large. However, it should be possible to change slightly the schemes (i.e. replace a message with a hash of a message), what will reduce the total transfer time, but requires to calculate hashes locally. After such modifications the schemes will be similar to signatures schemes used currently in a public key infrastructure, where in most of the cases it is only required to send a hash of a file to the cloud for outsourced verification.

The experiments show that using secure outsourcing (in one-malicious version of two untrusted program model) is only around 1.8 to 3.3 times slower (depending on the scheme) than outsourcing to a trusted cloud. The further acceleration of secure outsourcing algorithms would require secure outsourcing algorithms with minimal time for a local verification. Existing algorithms with efficiency ratio around 0.30 require hours to verify of 100k signatures.

The software implemented for the purpose of the experiments contain several simplifications that would have to be implemented in the production version (e.g., errors handling and boundary cases were not implemented as they were no needed in the experiments). Secure outsourcing accelerates computations, but also increases the total code complexity as expected. This is main practical drawback of that approach. However, when only computations of some operations (like BP and SM) are (securely) outsourced, then the additional code can be easily divided into software components that are easily manageable and reusable between different applications (e.g., one component for BP computation on a server can be easily reused between applications).

The future work include analysis of possibilities to speed up simultaneous verification of signatures created using schemes that use chameleon hashing, e.g., scheme *SIGN* [47]. Also, it would be interesting to investigate how techniques mentioned in this paper can be used to improve efficiency of verifiable searchable encryption schemes [48].

References

[1] S. S. Al-Riyami, K. G. Paterson, Certificateless public key cryptography, in: C.-S. Laih (Ed.), *ASIA CRYPT 2003*, Lecture Notes in Computer Science, Vol. 2894, Springer, 2003, pp. 452-473.

[2] W. Wu, Y. Mu, W. Susilo, X. Huang, Certificate-based Signatures Revisited, *Journal of Universal Computer Science*, Vol. 15, No. 8, pp. 1659-1684, April, 2009.

[3] X. Huang, Y. Mu, W. Susilo, D. S. Wong, W. Wu, Certificateless Signature Revisited, in: J. Pieprzyk, H. Ghodosi, E. Dawson (Eds.), *Australasian Conference on Information Security and Privacy 2007*, Lecture Notes in Computer Science, Vol. 4586, Springer, 2007, pp. 308-322.

[4] L. Zhang, F. Zhang, A New Provably Secure Certificateless Signature Scheme, *2008 IEEE International Conference on Communications*, Beijing, China, 2008, pp. 1685-1689.

[5] T. Hyla, J. Pejaś, A Hess-Like Signature Scheme Based on Implicit and Explicit Certificates, *Computer Journal*, Vol. 60, No. 4, pp. 457-475, March, 2017.

[6] H. Hacigumus, B. Iyer, S. Mehrotra, Providing Database As a Service, *Proceeding of the 18th International Conference on Data Engineering*, San Joes, CA, USA, 2002, pp. 29-38.

[7] X. Lei, X. Liao, T. Huang, F. Heriniaina, Achieving Security, Robust Cheating Resistance, and High-Efficiency for Outsourcing Large Matrix Multiplication Computation to a Malicious Cloud, *Information Sciences*, Vol. 280, pp. 205-217, October, 2014.

[8] J. Abawajy, Determining Service Trustworthiness in Intercloud Computing Environments, *2009 10th International Symposium on Pervasive Systems*, *Algorithms*, *and Networks*, Kaohsiung, Taiwan, 2009, pp. 784-788.

[9] F. Hess, Efficient Identity Based Signature Schemes Based on Pairings, *Proceeding of the International Workshop on Selected Areas in Cryptography 2002*, pp. 310-324, Springer, Berlin, Heidelberg, 2003.

[10] M. Girault, D. Lefranc, Server-Aided Verification: Theory and Practice, in: B. Roy (Ed.), *ASIA CRYPT 2005*, Lecture Notes in Computer Science, Vol. 3788, IACR, 2005, pp. 605-623.

[11] W. Wu, Y. Mu, W. Susilo, X. Huang, Provably Secure Server-aided Verification Signatures, *Computers and Mathematics with Applications*, Vol. 61, No. 7, pp. 1705-1723, April, 2011.

[12] W. Wu, Y. Mu, W. Susilo, X. Huang, Server-Aided Verification Signatures: Definitions and New Constructions, in: J. Baek, F. Bao, K. Chen, X. Lai (Eds.), *International Conference on Provable Security ProvSec 2008*, Lecture Notes in Computer Science, Vol. 5324, Springer-Verlag, 2008, pp. 141-155.

[13] B. Waters, Efficient Identity-based Encryption without Random Oracles, in: R. Cramer (Ed.), *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, Vol. 3494, Springer-Verlag, 2005, pp. 114-127.

[14] D. Boneh, G. Lynn, H. Shacham, Short Signature from The Weil Pairing, in: C. Boyd (Ed.), *Advances in Cryptology - ASIACRYPT 2001*, Lecture Notes in Computer Science, Vol. 2248, Springer-Verlag, 2001, pp. 514-532.

[15] S. S. M. Chow, M. H. Au, W. Susilo, Server-aided Signatures Verification Secure against Collusion Attack, *Information Security Technical Report*, Vol. 17, No. 3, pp. 46-57, February, 2013.

[16] Z. Qin, J. Sun, A. Wahaballa, W. Zheng, H. Xiong, Z. Qin, A Secure and Privacy-Preserving Mobile Wallet with Outsourced Verification in Cloud Computing, *Computer Standards & Interfaces*, Vol. 54, Part 1, pp. 55-60, November, 2017.

[17] B. Chevallier-Mames, J. S. Coron, N. McCullagh, D. Naccache, M. Scott, Secure Delegation of Elliptic-Curve Pairing, in: D. Gollmann, J.-L. Lanet, J. Iguchi-Cartigny (Eds.), *Smart Card Research and Advanced Application Conference 2010*, Lecture Notes in Computer Science, Vol. 6035, Springer, 2010, pp. 24-35.

[18] S. Canard, Delegating a Pairing Can Be both Secure and Efficient, in: I. Boureanu, P. Owesarski, S. Vaudenay (Eds.), *International Conference on Applied Cryptography and Network Security 2014*, Lecture Notes in Computer Science, Vol. 8479, Springer International Publishing, 2014, pp. 549-565.

[19] X. Chen, W. Susilo, J. Li, D. S. Wong, J. Ma, S. Tang, Q. Tang, Efficient algorithms for Secure Outsourcing of Bilinear Pairings, *Theoretical Computer Science*, Vol. 562, No. C, pp. 112-121, January, 2015.

[20] S. Hohenberger, A. Lysyanskaya, How To Securely Outsource Cryptographic Computations, in: J. Kilian (Ed.), *Second Theory of Cryptography Conference*, Lecture Notes in Computer Science, Vol. 3378, Springer, 2005, pp. 264-282.

[21] H. Tian, F. Zhang, K. Ren, Secure Bilinear Pairing Outsourcing Made More Efficient and Flexible, *Proceedings of the 10th ACM Symposium on Information*, *Computer and Communications Security*, Singapore, Republic of Singapore, 2015, pp. 417-426.

[22] M Dong, Y. Ren, X. Zhang, Fully Verifiable Algorithm for Secure Outsourcing of Bilinear Pairing in Cloud Computing, *KSII Transactions on Internet and Information Systems*, Vol. 11, No. 7, pp. 3648-3663, July, 2017.

[23] M. Dong, Y. L. Ren, Efficient and Secure Outsourcing of Bilinear Pairings with Single Server, *Science China Information Sciences*, Vol. 61, No. 3, 039104, 2018.

[24] A. Guillevic, D. Vergnaud, Algorithms for Outsourcing Pairing Computation, in: M. Joye, A. Moradi (Eds.), *International Conference on Smart Card Research and Advanced Applications 2014*, Lecture Notes in Computer Science, Vol. 8968, Springer, 2014, pp. 193-21.

[25] Y. Luo, S. Fu, K. Huang, D. Wang, M. Xu, Securely Outsourcing of Bilinear Pairings with Untrusted Servers for Cloud Storage, *IEEE TrustCom/BigDataSE/ISPA*, Tianjin, China, 2016, pp. 623-629.

[26] X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New Algorithms for Secure Outsourcing of Modular Exponentiations, in: S. Foresti, M. Yung, F. Martinelli (Eds.), *European Symposium on Research in Computer Security 2012*, Lecture Notes in Computer Science, Vol. 7459, Springer, 2012, pp. 541-556.

[27] X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New Algorithms for Secure Outsourcing of Modular Exponentiations, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 9, pp. 2386-2396, September, 2014.

[28] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. M. Chow, Z. Liu, X. Tan, Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage, in: M. Kutyłowski, J. Waidya (Eds.), *European Symposium on Research in Computer Security 2014*, Lecture Notes in Computer Science, Vol. 8712, Springer, 2014, pp. 326-343.

[29] Y. Ding, Z. Xu, J. Ye, K.-K. Raymond Choo, Secure Outsourcing of Modular Exponentiations under Single Untrusted Programme Model, *Journal of Computer and System Sciences*, Vol. 90, pp. 1-13, December, 2017.

[30] P. Q. Nguyen, I. E. Shparlinski, J. Stern, Distribution of Modular Sums and the Security of the Server Aided Exponentiation, in: K. Y. Lam, I. Shparlinski, H. Wang, C. Xing (Eds.), *Cryptography and Computational Number Theory. Progress in Computer Science and Applied Logic*, Vol. 20, Birkhauser Verlag, 2001, pp. 331-342.

[31] M.V. Dijk, D. Clarke, B. Gassend, G. E. Suh, S. Devadas, Speeding up Exponentiation Using an Untrusted

Computational Resource, *Designs*, *Codes and Cryptography*, Vol. 39, No. 2, pp. 253-273, May, 2006.

[32] K. Zhou, J. Ren, Secure Outsourcing of Scalar Multiplication on Elliptic Curves, *IEEE ICC 2016 Communication and Information Systems Security Symposium*, Kuala Lumpur, Malaysia, 2016, pp. 1-5.

[33] Certicom Research, *SEC 2: Recommended Elliptic Curve Domain Parameters*, Version 2.0, http://www.secg.org/sec2-v2.pdf, 2010.

[34] G. M. Zaverucha, D. R. Stinson, Group Testing and Batch Verification, in: K. Kurosawa (Ed.), *International Conference on Information Theoretic Security 2009*, Lecture Notes in Computer Science, Vol. 5973, Springer-Verlag, 2010, pp. 140-157.

[35] J. Camenisch, S. Hohenberger, M. Pedersen, Batch Verification of Short Signatures, in: M. Naor (Ed.), *EURO CRYPT 2007*, Lecture Notes in Computer Science, Vol. 4515, International Association for Cryptology Research, 2007, pp. 246-263.

[36] A. L. Ferrara, Practical Short Signature Batch Verification, in: M. Fischlin (Ed.), *Cryptographers Track RSA Conference 2009*, Lecture Notes in Computer Science, Vol. 5473, Springer-Verlag, 2009, pp. 309-324.

[37] C.I. Fan, P.-H. Ho, J.-J. Huang, Y.-F. Tseng, Secure Certificateless Signature Scheme Supporting Batch Verification, *2013 Eighth Asia Joint Conference on Information Security*, Seoul, South Korea, 2013, pp. 8-11.

[38] T. Cao, D. Lin, R. Xue, Security Analysis of Some Batch Verifying Signatures from Pairings, *International Journal of Network Security*, Vol. 3, No. 2, pp. 138-143, September, 2006.

[39] A. Fiat, Batch RSA, *Journal of Cryptology*, Vol. 10, No. 2, pp. 75-88, March, 1997.

[40] L. Harn, Batch Verifying Multiple RSA Digital Signatures, *Electronics Letters*, Vol. 34, No. 12, pp. 1219-1220, June, 1998.

[41] H. Yoon, J. H. Cheon, Y. Kim, Batch Verifications with ID-Based Signatures, *Proceedings of International Conference on Information Security and Cryptology 2004*, Seoul, Korea, 2004, pp. 233-248.

[42] C. Shi, D. Pu, W. C. Choong, An Efficient Identity-Based Signature Scheme with Batch Verifications, *InfoScale '06 Proceedings of the 1st international conference on Scalable information systems*, Hong Kong, China, 2006, pp. 1-6.

[43] M. Geng, F. Zhang, Batch Verification for Certificateless Signature Schemes, *Proceedings of the International Conference on Computational Intelligence and Security 2009*, Beijing, China, 2009, pp. 288-292.

[44] C.-I. Fan, P.-H. Ho, Y.-F. Tseng, Strongly Secure Certificateless Signature Scheme Supporting Batch Verification, *Mathematical Problems in Engineering*, Vol. 2014, Article ID 854135, doi:10.1155/2014/854135, 2014.

[45] X. Qin, S. Zhang, L. Jia, Research on Pairing-Based Batch Verification, *2010 International Conference on Communications and Mobile Computing*, Shenzhen, China, 2010, pp. 46-50.

[46] CertiVox/MIRACL, https://github.com/CertiVox/MIRACL.

[47] X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, K. Kim, Identity-based Chameleon Hashing and Signatures without Key Exposure, *Information Sciences*, Vol. 265, pp. 198-210, May, 2014.

[48] Z. Liu, T. Li, P. Li, C. Jia, J. Li, Verifiable Searchable Encryption with Aggregate Keys for Data Sharing System, *Future Generation Computer Systems*, Vol. 78, pp. 778-788, January, 2018.

Biography

Tomasz Hyla received Ph.D. degree in Computer Science, Cryptography in 2011 from West Pomeranian University of Technology in Szczecin, Poland. Currently, he is employed as Assistant Professor and Head of Information Security Research Group. His main research theme is pairing-based cryptography with an emphasis on new digital signature schemes.