entropy

MDPI

Article

Revocable Signature Scheme with Implicit and Explicit Certificates

Jerzy Pejaś [1,†], Tomasz Hyla [1,*,†] and Wojciech Zabierowski [2,†]

[1] Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, 71-210 Szczecin, Poland; jerzy.pejas@zut.edu.pl
[2] Department of Microelectronics and Computer Science, Lodz University of Technology, 93-005 Lodz, Poland; wojciech.zabierowski@p.lodz.pl
* Correspondence: thyla@zut.edu.pl
† These authors contributed equally to this work.

Abstract: This paper addresses the certificate revocation problem and proposes the first revocable pairing-based signature scheme with implicit and explicit certificates (IE-RCBS-kCAA). We should no longer discuss whether to revoke certificates but how to do it effectively, ensuring both the scalability of the revocation operation and the non-repudiation of the signature in the short or long term. Under the computational difficulty assumptions of the modified collusion attack algorithm with k traitors (k-mCAA) and discrete logarithm (DL) problems, we demonstrate that our scheme is secure against existential unforgeability under chosen message attacks (EUF-IERCBS-kCAA-CMA) in a random oracle model. The proposed solution is scaled and allows the use of many trusted status authorities that issue explicit short-term certificates confirming the validity of explicit long-term certificates. Furthermore, we demonstrate that our signature scheme has a short-term non-repudiation property for the shell validity model.

Keywords: signature scheme; implicit and explicit certificates-based public key cryptography; bilinear pairing; revocation; non-repudiation

Citation: Pejaś, J.; Hyla, T.; Zabierowski, W. Revocable Signature Scheme with Implicit and Explicit Certificates. *Entropy* **2023**, *25*, 1315. https://doi.org/10.3390/e25091315

Academic Editor: Boris Ryabko

Received: 20 June 2023
Revised: 31 August 2023
Accepted: 7 September 2023
Published: 9 September 2023

1. Introduction

Digital signatures are a critical component that ensure the integrity, authenticity and non-repudiation of electronic documents. In public key cryptography (PKC), a digital signature is considered valid if it is mathematically correct and a related certificate is valid. The certificates are issued for a limited period (usually two years), they can be revoked (e.g., a signer's private key is compromised) and in most cases, a user certificate is issued by an intermediate CA (certificate authority), and other certificates are issued by another intermediate CA or by a root CA with a self-signed certificate. The root CA is considered a trust anchor. All certificates between an end-entity certificate and a trust anchor certificate form the trusted certificate path. The validation of this chain is a challenging task.

Three models for certificate validation exist [1]. In the first one, called the *shell model*, the certificate is valid as long as all the certificates in the certification path are valid when the signature is verified. The second one is the *modified shell model*. In this model, the signing time is the basis for decision making about the signature validity (this requires a timestamp). The third model is the *chain model*: once a signature is valid at signing time, it remains valid all the time. The chain model can only provide non-repudiation of property over a long period of time [1,2].

The purpose of certificate validation is to confirm the authenticity of the public key, i.e., to provide proof that the public key belongs to the signer and that the associated private key was under the sole control of an owner at the moment of signature creation. The process of verifying certificates for signatures that need to be valid for a long period of time is one of the main difficulties when implementing and managing a public key infrastructure. The

authenticity of a public key in a public key cryptosystem can be achieved in two ways: either explicitly or implicitly. During explicit authentication, the public key authenticity can be verified directly using the certificate issued by a CA. In implicit authentication, the secret certificate issued by a CA (i.e., partial or complete private key) can be verified indirectly during signature verification or decryption operation. Combining both approaches and obtaining cryptographic schemes based on explicit and implicit certificates is another possible solution (e.g., T. Hyla et al. [3]).

It seemed that the certificate validation problem had been solved in 1984 when Shamir [4] introduced a new identity-based public key cryptography (ID-PKC) concept utilizing a user's publicly known identity information as her/his public key. Since that time, many schemes based on that concept have been proposed. First came certificateless public key cryptography (CL-PKC), which removed the key escrow problem, one of the major problems in ID-PKC. Next, (implicit) certificate-based public key cryptography (IC-PKC) schemes were proposed. Compared to CL-PKC, IC-PKC schemes are resistant to public key replacement attacks. The certificate is implicit because it must be kept secret and is not used directly during signature verification. In other words, the implicit certificate is a partial secret key sent to the user by a trusted authority (TA) that, in comparison to CL-PKC schemes, additionally binds the user's identity with its public key and the parameters of the trusted authority. Note that some authors do not use the word "implicit", which could be misleading as someone might think that certificates are always public.

The great interest shown in the ID-PKC, CL-PKC and IC-PKC schemes is because they eliminate explicit certificates from the encryption or signature schemes and allow the need to manage the status of certificates to be dispensed with. In practice, this is not the case, as the problem of certificate status management is shifted to the user identity management level [5,6]. In addition, this generates problems with public key distribution, such as vulnerability to public key replacement attacks. As a result, key invalidation in ID-PKC, CL-PKC and IC-PKC schemes can be even more cumbersome than in traditional PKC-based cryptosystems.

1.1. Related Works

The digital signature schemes must include a revocation mechanism to support non-repudiation and achieve Girault trust level 3 (see Girault, M. [7]). The revocation mechanism in identity-based certificates or based on implicit certificate schemes is implemented through a few techniques: using an online mediator, periodically updating the user's secret using a secret channel and using time keys that can be sent using a public channel.

In the online mediator (SEM, Security Mediator) technique [8,9], a user's partial private key is divided into two parts. One part is sent to the user and one to the SEM. The advantage of this approach is instantaneous revocation. The main drawbacks of this approach are the need for a secure (confidential) channel, and that the user cannot create a signature independently. In addition, the SEM must store a large amount of partial secret keys. Secondly, it is possible to generate private keys over regular periods [10–12]. When revocation of users is needed, the trusted authority (TA) just stops updating users' partial private keys. The main drawback is the need for secure channels between users and TA.

In 2001, Boneh and Franklin [13] proposed a method in which a trusted private key generator (PKG) periodically updates private keys for all non-revoked users. Boldyreva et al. [14] introduced the first scalable revocable identity-based encryption scheme. The scheme was later improved by Libert and Vergnaud [15] and by Seo and Emura [16]. In 2012, Tseng and Tsai proposed efficient revocable identity-based encryption [17] and signature schemes [18]. They used a different method of private key construction, where the private key consists of a fixed initial private key and a time key, which is issued periodically by the PKG for non-revoked users. The key can be sent using a public channel. Their work was later reviewed by [19]. Chen et al. [20] proposed a selective-ID secure revocable identity-based encryption (RIBE) scheme from lattices. Cheng et al. [21] presented an

adaptive-ID version of the [20] scheme. Lee et al. in [22,23] constructed a RIBE scheme based on pairings using the subset difference method.

In 2014, Sun et al. [24] proposed a revocable certificateless signature (RCLS) scheme in which the TA produces an initial partial private key and a time key corresponding to each period. The time key is transmitted over a public channel. Next, Sun and Shen [25] and Sun et al. [26] proposed a revocable certificateless signature (RCLS) scheme without the use of bilinear pairings.

In 2017, Jia et al. [27] proposed an efficient revocable identity-based signature (RIBS) scheme in which the revocation functionality is outsourced to a cloud revocation server. In their solution, a short-term key (time key) is issued by the cloud server instead of KGC. Recently, a similar solution based on semi-trusted cloud revocation agents (s-CRAs) was used by Ma et al. [28].

1.2. Motivation and Contribution

This paper proposes a new revocable signature IE-RCBS-kCAA scheme with implicit and two explicit (short- and long-term) certificates, which is secure in the random oracle model under the hardness assumption of the *modified k-CAA* problem and the discrete logarithm (DL) problem. When an explicit long-term certificate is revoked, its status information (in the form of an explicit short-term certificate) is available online at the moment of signature generation. The method used for revoking and providing certificate status is similar to that presented by Yum et al. [29].

The implicit and short-term explicit certificates are two components of the signature key that can be used to sign documents. A signatory who wishes to reject the signatures he or she has generated may intentionally compromise his or her signature key and falsely claim that the signatures have been forged by someone else. Such a scenario is impossible with an explicit short-term certificate because the signer cannot revoke the explicit short-term certificate used as part of the full signature key.

Using explicit short-term certificates ensures that revocation latency (i.e., a time lag between revoking the certificate and informing the relaying parties) is irrelevant, as the digital signature is valid only until the associated short-term explicit certificate expires. Because of the proposed signature validation approach based on the certificate validation shell model (see Section 1), the acceptance of such a signature should not pose any risk to a relaying party.

In our approach, a user's private key consists of a secret value, a long-term partial private key and a short-term explicit certificate (a time key). The first two partial keys are kept secret by the user, and their authenticity is confirmed by a long-term certificate issued by the trusted authority (TA). An explicit short-term certificate is periodically updated and sent over a public channel. To perform such an operation (unlike the cloud-based revocation server proposed by Jia et al. in Figure 1), we suggest using the trusted status authority (TSA), which issues a new short-term explicit certificate for each valid long-term explicit certificate and stops doing so when the long-term explicit certificate is revoked.

The IE-RCBS-kCAA scheme fulfils the following conditions:

1. An explicit short-term certificate eliminates the need to generate CRLs used in traditional PKI systems; furthermore, it serves as non-repudiation evidence of a digital signature;
2. During the signature creation process, a three-component user's private key is used; this approach allows Girault's trust level 3 security to be achieved; only the verification process, in addition to the public key and explicit signer's certificates, indirectly references other parties' keys, including TA keys;
3. A signer's public key, as in the related two partial private keys, contains three component groups: the signer generates the first, while two others are created by trusted and trusted status authorities;

4. The short- and long-term explicit certificates of a signer are public, i.e., these certificates are used in the signature verification process and to verify their authenticity and their validity;
5. A signature verification process uses short- and long-term explicit certificates, where explicit short-term certificates play a role in the certificate status;
6. The strongest security property for digital signatures is provided, i.e., existential enforceability against adaptively chosen message attacks.

Additionally, in the scheme, a user can only change his public and private keys with TA acceptance and vice versa. A TA cannot generate a false private key of any user to forge a signature without being detected by the user. Hence, the scheme fulfils Girault's level 3 security requirement.

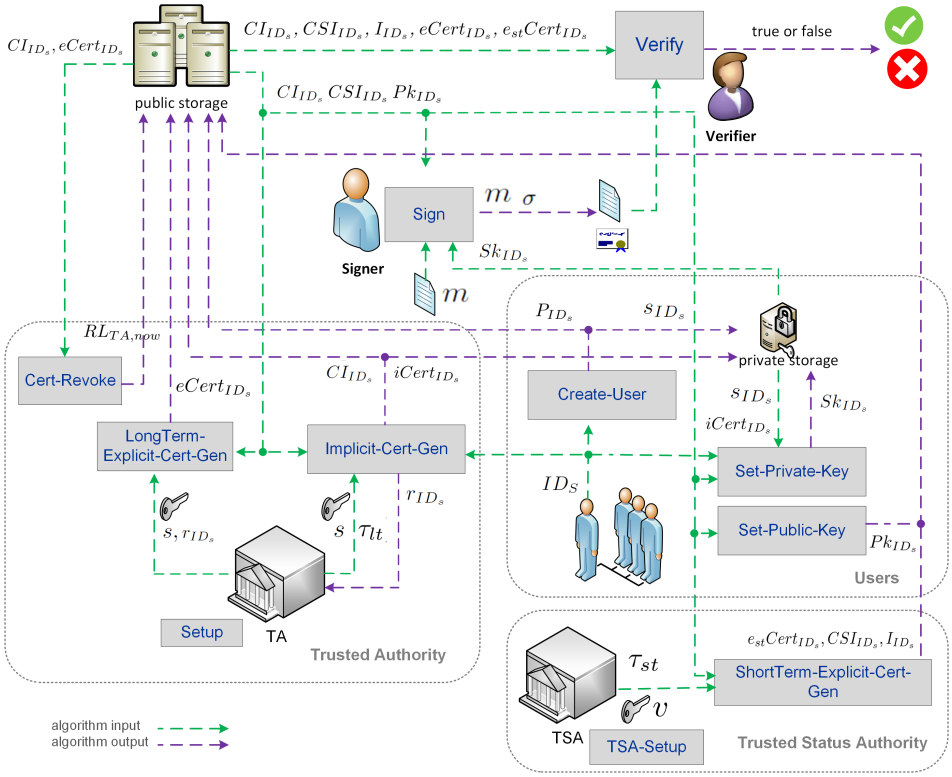

Figure 1. Architecture framework of IE-RCBS-kCAA signature scheme.

1.3. Paper Organisation

In addition to the introductory section, this paper consists of four sections and two appendices. Section 2 introduces the concept of a signature scheme based on implicit and explicit certificates and its security model against three different attack types. Section 3 proposes a randomized IE-RCBSS-kCAA pairing-based signature scheme, and Section 4 analyses its security in a random oracle. Section 5 presents the analysis of the scheme's efficiency. The paper ends with conclusions.

2. Signature Scheme Framework Architecture and Its Security Model

This section describes the architecture framework of our IE-RCBS-kCAA signature scheme and its security model. The definitions of asymmetric bilinear map groups and the hard computational problems (discrete logarithm (DL) problem, kCAA problem, k-mCAA problem) can be found in Hyla et al. [30] and Mitsunari et al. [31].

2.1. Signature Scheme Framework

The architecture framework of the IE-RCBS-kCAA signature scheme is shown in Figure 1. This architecture involves three parties: the trusted authority (TA), the trusted

status authority (TSA) and the users (signers and verifiers). At system initialization, the TA generates and publishes common parameters. The TSA can use these parameters to generate its secret master private status key or independently generate its parameters and then use them to generate the secret master private status key. This solution allows the TA and TSA to work in the first case in the same algebraic groups and the second in two different, independent ones.

Next, the TA issues the partial secret key for each registered user using its master system key. The TSA is an authentication service that decides the validity of an explicit long-term certificate at the current time and issues an explicit short-term certificate if the TSA answer is positive. Therefore, on request, the TSA checks the signer's long-term explicit certificate status according to the signed revocation list from the TA. The revocation list issued at time t by a trusted authority, TA, is denoted $RL_{TA,t}$. This list contains the indexes of revoked long-term certificates and is updated periodically. Suppose a signer's long-term certificate is in the signed revocation list. In that case, the TSA can send back a revoked response with a long-term explicit certificate status value equal to *revoked* or refuse the request. Otherwise, it outputs *good* or *unknown*. This last state indicates that the TSA does not know the certificate being requested.

The trust model (understood as building trust relationships between cryptographic keys and their owner's identity) with the TSA, and separating its role from the TA role, allows various business models to be obtained that can provide certificate issuing services (explicit or implicit) and determine their status. In a particular case, the TSA may act as an independent third party and provide certificate status verification services to multiple TA authorities. This type of scenario aligns with the model that has long been used in PKI systems: CAs issue certificates, and the OCSP server issues certificates of their status [32].

Let us assume that the earlier mentioned trust authorities and trusted status authorities are part of the trust model based on a common set of system parameters. These parameters may include, among others, the same algebraic groups G_1, G_2 and G_T. It is easy to notice then that such an assumption allows us to effectively solve the scalability problem of the trust architecture shown in Figure 1 and to adapt it to many users. In that case, users can use not only many TSAs but also many TAs. However, this paper only considers the trust model with a single TA and TSA.

The IE-RCBS-kCAA consists of eleven polynomial-time algorithms (compare T. Hyla et al. [33]): the system initialization algorithm (TA-Setup), TSA initialization algorithm (TSA-Setup), secret user's key generation algorithm (Create-User), implicit certificate extraction algorithm (Implicit-Cert-Gen), long-term explicit certificate generation algorithm (LongTerm-Explicit-Cert-Gen), short-term explicit certificate algorithm (ShortTerm-Explicit-Cert-Gen), full user's private key creation algorithm (Set-Private-Key), full user's public key creation algorithm (Set-Private-Key), long-term certificate revocation algorithm (Cert-Revoke) signing algorithm (Sign) and verification algorithm (Verify).

Definition 1 (IE-RCBS-kCAA scheme)**.** *An implicit and explicit certificates-based signature against k-traitors collusion attack scheme consists of the following eleven polynomial-time algorithms:*

TA-Setup $(1^k) \rightarrow (s, params)$. *A security parameter* 1^k *is an input and outputs the certifier's master private key* s*, the system parameters are* $params$ *and a revocation list* $RL_{TA,thisUpdate}$ *(initially empty), where* $thisUpdate$ *indicates the issue date of this RL, which are then properly distributed in the system. The TA runs the algorithm and, when completed, keeps the master private key* s *secret, while the RL and* $params$ *are publicly available to the TSA and all other users on the system, respectively.*

The TA runs the algorithm and, in secret, keeps the master private key s*, while the RL and* $params$ *are publicly accessible to the TSA and all other users in the system, respectively.*

TSA-Setup $(params) \rightarrow (v, V_0, T_0)$. *The algorithm takes as input the system parameters* $params$ *and outputs a master private status key* v *and two related TSA public keys* (V_0, T_0)*.*

Create-User $(params, ID_s) \rightarrow (s_{ID_s}, P_{ID_s})$. *The user runs the algorithm, and the input is the system parameters and the signer's identity. The output is the user's secret key value* s_{ID_s} *and the corresponding first partial public key* P_{ID_s}.

Implicit-Cert-Gen $(params, s, ID_s, P_{ID_s}, \tau_{lt}) \rightarrow (CI_{ID_s}, iCert_{ID_s}, r_{ID_s})$. *This algorithm takes as input the system parameter params, master private key s, the identity* ID_s *of a user, its first partial public key* P_{ID_s} *and a certificate validity period* τ_{lt}. *It outputs the user's certificate information* CI_{ID_s}, *an implicit certificate* $iCert_{ID_s}$ *and the secret key* r_{ID_s} *used by the TA during the user's implicit and explicit certificates' generation that is unknown to this user. The TA runs the algorithm once for each user, and the corresponding implicit certificate is distributed to the user secretly.*

LongTerm-Explicit-Cert-Gen $(params, s, CI_{ID_s}, r_{ID_s}, q_{ID_s}) \rightarrow (eCert_{ID_s})$. *The input is the system parameter params, master private key s, the user's certificate information* CI_{ID_s}, *the secret key* r_{ID_s} *related to the user's implicit and explicit certificates, and the hash value* q_{ID_s}. *The output is an explicit long-term certificate* $eCert_{ID_s}$ *that is sent to the user by a public channel. A TA runs this algorithm once for each user.*

ShortTerm-Explicit-Cert-Gen $(params, v, V_0, T_0, bstr, CI_{ID_s}, eCert_{ID_s}, \tau_{st}) \rightarrow (e_{st}Cert_{ID_s}, CSI_{ID_s}, I_{ID_s})$. *This algorithm takes as input the system parameter params, a master private status key v and two related TSA public keys (*V_0, T_0*), the bitstring bstr (e.g., related with the signed message),* CI_{ID_s} *and his/her long-term explicit certificate, and a period* τ_{st}. *The TSA first checks the current* $RL_{TA,t}$. *If the request concerns the non-revoked long-term explicit certificate, then the TSA outputs an explicit short-term certificate* $e_{st}Cert_{ID_s}$, *the certificate status information* CSI_{ID_s} *and auxiliary public information* I_{ID_s} *that is sent to the user by a public channel. A TSA runs this algorithm once for each user's request.*

Set-Private-Key $(params, CI_{ID_s}, CSI_{ID_s}, s_{ID_s}, iCert_{ID_s}, e_{st}Cert_{ID_s}) \rightarrow Sk_{ID_s}$. *The user runs this algorithm. The algorithm takes as input the system parameters params, user's certificate information* CI_{ID_s}, *the certificate status information* CSI_{ID_s}, *a secret key* s_{ID_s}, *an implicit certificate* $iCert_{ID_s}$ *and a short-term explicit certificate* $e_{st}Cert_{ID_s}$, *and returns the corresponding full user's private key* $Sk_{ID_s} = (s_{ID_s}, iCert_{ID_s}, e_{st}Cert_{ID_s})$.

Set-Public-Key $(params, CI_{ID_s}) \rightarrow Pk_{ID_s}$: *the user S run the algorithm with the certificate information* CI_{ID_s}. *It returns the full long-term public key in the form* $Pk_{ID_s} = (P_{ID_s}, R'_{ID_s}, R'_{ID_s})$.

Cert-Revoke $(params, CI_{ID_s}, eCert_{ID_s}) \rightarrow RL_{TA,thisUpdate}$: *for an input tuple* $(CI_{ID_s}, eCert_{ID_s})$ *with the explicit long-term certificate* $eCert_{ID_s}$ *that is requested to be revoked, the TA verifies entity credentials, and if the entity is authorized successfully, then the TA revokes the certificate and places it on the signed revocation list* $RL_{TA,thisUpdate}$ *that is issued at thisUpdate.*

Sign $(params, m, CI_{ID_s}, Sk_{ID_s}, Pk_{ID_s}) \rightarrow \sigma$. *The signer runs the* ***Sign*** *algorithm that generates a signature* σ *for the given input: the params, a message m, a user certificate information* CI_{ID_s} *and the user's full key pair* (Sk_{ID_s}, Pk_{ID_s}).

Verify $(params, (m, \sigma), CI_{ID_s}, CSI_{ID_s}, I_{ID_s}, eCert_{ID_s}, e_{st}Cert_{ID_s}) \rightarrow \{true, false\}$. *Everyone can run the algorithm Verify to check the validity of a signature. Taking as input a message/signature pair* (m, σ), *a user's certificate information* CI_{ID_s}, *a certificate status information* CSI_{ID_s}, *an auxiliary public information* I_{ID_s}, *and long- and short-term explicit certificates (*$eCert_{ID_s}$, $e_{st}Cert_{ID_s}$*), it outputs true when* σ *is a valid signature. Otherwise, it outputs* $false$.

It is required that if σ = Sign ($params$, m, CI_{ID_s}, Sk_{ID_s}, $eCert_{ID}$) then Verify ($params$, m, σ, CI_{ID_s}, CSI_{ID_s}, I_{ID_s}, $eCert_{ID_s}$, $e_{st}Cert_{ID_s}$) = $true$, where the public parameters $params$, the signer's private/public key pair (Sk_{ID_s}, Pk_{ID_s}) and the long- and short-term explicit certificates ($eCert_{ID_s}$, $e_{st}Cert_{ID_s}$) are generated based on the specification of the algorithms: TA-Setup, TSA-Setup, Create-User, Implicit-Cert-Gen, LongTerm-Explicit-Cert-Gen and ShortTerm-Explicit-Cert-Gen.

Remark 1. *Implicit-Cert-Gen and Explicit-Cert-Gen algorithms are successful when the TA positively verifies the identity and certificate information of CI_{ID_s} confirming this identity. Furthermore, whenever a user requests a certificate for a public key P_{ID_s}, the user must prove the possession of the corresponding secret key s_{ID_s} to the certifier, similar to a traditional public key infrastructure. Similar remarks apply to ShortTerm-Explicit-Cert-Gen: a positive result of this algorithm is only returned for the associated valid long-term unencrypted certificate.*

2.2. Security Model

The security proof of the proposed IE-RCBS-kCAA signature scheme is based on the commonly accepted standard security notion EUF-CMA (existential unforgeability under chosen message attack). The EUF-CMA notation guarantees the highest security level of the signature scheme and thus the resistance of the signature scheme against the strongest attacks of the adversary.

The security proofs is a claim made within the random oracle model, where a hash function finally replaces the random oracle. It is easy to see that the last step is heuristic in nature. In practice, the heuristics are successfully used for problem solving (e.g., [34–37]). However, the security proof in the oracle model can only be treated as a heuristic argument for the security of the cryptographic scheme, but without a guarantee for the security of its real implementation (Bellare and Rogaway [38]).

For the IE-RCBS-kCAA signature scheme, four cases of access or lack of access by an adversary to TA and TSA master keys should be considered:

(a) An adversary does not know the TA and TSA master keys;
(b) An adversary knows the TA and TSA master keys;
(c) An adversary knows the TA master private key and does not know the TSA master private status key;
(d) An adversary does not know the TA master private key TA and knows the TSA master private status key TSA.

Access to or lack of access to TA and TSA keys may depend on the adversary's knowledge or ignorance of different user keys. Consequently, this allows us to define five different types of adversaries, the capabilities of which are shown in Table 1. Each type of adversary has its role and access rights (yes/no) to the user's secrets or public key replacement. For example, the A_1 adversary is a user who has not yet been registered and does not have a certificate. The purpose of the adversary attack is to impersonate this type of user and forge his/her signature. It is assumed that the adversary does not have access to the TA and TSA master keys and to the implicit certificate of the target user but has access to his/her short-term explicit certificate, the secret key, and can change his public key.

Note that even if the A_1, A_3 and A_4 adversaries cannot access the TSA's master private status key, it still provides them access to the explicit short-term certificate. The TSA is fair and acts as an oracle, responding to any correct requests unless they concern a revoked certificate. In the latter case, the adversary does not receive a valid short-term explicit certificate for the next period. However, acting as a user with the revoked implicit certificate, they can collude with other legal users and generate its correct value.

A thorough analysis of the adversary types and their capabilities in Table 1 shows that the adversaries A_1 and A_5 and A_2 and A_4 have equivalent capabilities to falsify the target user's signature. Hence, in the case of the proposed signature scheme with two trust authorities (TA and TSA), only three types of adversaries (A_1, A_2 and A_3), should be considered. As a result, the security model is similar to the models proposed for invalidation signature schemes with a single trust authority (see, e.g., Y. Sun et al. [24] and Y. Huang et al. [39]).

Table 1. The adversary types with different capabilities.

Adversary Type	TA Master Key	TSA Master Key	Implicit Certificate	Short-Term Explicit Certificate	User's Secret Key	Public Key Replacement
A_1 (*non-certified user*)	no	no	no	yes	yes	yes
A_2 (*certified user*)	yes	yes	yes	yes	no	no
A_3 (*user with revoked certificate*)	no	no	yes	no	yes	yes
A_4 (*certified user*)	yes	no	yes	yes	no	no
A_5 (*non-certified user*)	no	yes	no	yes	yes	yes

Based on the above comments, the security model of the proposed IE-RCBS-kCAA scheme, from now on referred to as EUF-IERCBS-kCAA-CMA, is defined by three games between challenger C and adversary A, assuming that the adversary chooses which game to play. In all cases, adversary $A = (A_1, A_2, A_3)$ is trying to break the EUF-CMA security of the IE-RCBS-kCAA scheme, i.e., the formal model describing existential unforgeability. We use two types of adversaries with different capabilities: Type I adversary and Type II adversary (see, e.g., [3]) to describe the first two games. For the third type of adversary, i.e., Type III adversary, we adopt the security notation introduced by Y. Sun et al. [24] and Y. Huang et al. [39] that is necessary for the security proofs to come. Type I and II adversaries are similar to those defined in [30] and their descriptions are omitted here. Type III adversary (A_3) represents a revoked certified user whose long-term explicit certificate is no longer valid. However, it should be noted that a revoked user still holds her/his implicit certificate and related secret key. However, the TSA stops issuing the subsequent short-term explicit certificates to her/him. The adversary cannot gain the TA's master secret keys and the TSA's master private status key but can replace the public key of any user, except the target user, with a value of her/his choice. The security model categorises potential adversaries based on their attack capabilities and classifies Type I/II/III adversaries into three categories (see Li, J., et al. [40–42] and Huang, X., et al. [43]): Normal adversary, Strong adversary and Super adversary. The scheme should resist a Super Type I/II/III adversary (in Games I/II/III), who can obtain a valid signature under the public key chosen by itself without providing the corresponding secret.

Definition 2. *An implicit and explicit certificate revocable signature scheme IE-RCBS-kCAA has existential unforgeability against chosen message attacks (EUF-IERCBS-kCAA-CMA) if no probabilistic polynomial-time adversary has a non-negligible probability of winning Game I, Game II and Game III.*

3. A Novel Revocable Implicit and Explicit Certificates-Based Signature Scheme

3.1. The Revocable Signature Scheme with Common System Parameters (IE-RCBS-kCAA)

The IE-RCBS-kCAA scheme consists of eleven polynomial-time algorithms: TA-Setup, TSA-Setup, Create-User, Implicit-Cert-Gen, LongTerm-Explicit-Cert-Gen, Set-Private-Key, Cert-Revoke, Get-Cert-Status, Sign and Verify. The algorithms are as follows.

1. **TA-Setup**: The system parameters are $params = \{G_1, G_2, G_T, p, \hat{e}, P, P_0, Q, Q_0, H_1, H_2, H_3\}$, where $|G_1| = |G_2| = |G_T| = p$ for some prime number $p \geq 2^k$ (k is the system security number), (P, Q) are generators of, respectively, G_1 and G_2 such that $\hat{e}(P, Q) = g$, $P_0 = sP$ and $Q_0 = sQ$, the system's master public keys with the master private key $s \in Z_p^*$, $H_1, H_2 : \Gamma \rightarrow Z_p$ and $H_3 : \{0,1\}^* \rightarrow Z_p$ are three secure cryptographic hash functions. Γ means a string space that defines a user with the identity ID. When ID contains more information other than the identity, we mark it as CI or CSI.
2. **TSA-Setup** (*params*): The TSA chooses a random number $v \in Z_p^*$ as its master private status key and calculates its public keys $V_0 = vP$ and $T_0 = vQ$.

3. **Create-User** $(params, ID_s)$: The user ID_s chooses a random number $s_{ID_s} \in Z_p^*$, sets s_{ID_s} as the secret key and produces the corresponding first partial long-term public key $P_{ID_s} = s_{ID_s}P$. The secret key s_{ID_s} is kept secret, while the user sends P_{ID_s} to the TA over an authenticated channel.
4. **Implicit-Cert-Gen** $(params, s, ID_s, P_{ID_s}, \tau_{lt})$: Given ID_s presenting S's identity, his partial long-term public key P_{ID_s} and a period τ_{lt}, the trust authority TA:
   (a) Randomly selects $r_{ID_s} \in Z_p^*$ and computes respective second and third partial long-term public keys $(R'_{ID_s}, R''_{ID_s}) = (r_{ID_s}P, r_{ID_s}Q)$;
   (b) Composes the user's certificate information CI_{ID_s}, including the TA's public keys (P_0, Q_0), identifiers ID_s and ID_{TA} of the user S and the TA, respectively, first, second and third partial public keys $(P_{ID_s}, R'_{ID_s}, R''_{ID_s})$, and the period τ_{lt} for which the information CI_{ID_s} is valid;
   (c) For P_{ID_s} and (R'_{ID_s}, R''_{ID_s}) computes:
   $$q_{ID_s} = H_1(CI_{ID_s}) \tag{1}$$
   (d) Generates S's partial private key (an implicit certificate):
   $$iCert_{ID_s} = \frac{1}{s + r_{ID_s} q_{ID_s}} Q \tag{2}$$
   and transmits it to the user S secretly; in addition, TA sends CI_{ID_s}.
5. **LongTerm-Explicit-Cert-Gen** $(params, s, CI_{ID_s}, r_{ID_s}, q_{ID_s})$: The TA generates the signer's S explicit certificate using parameters provided by S and the values created when executing the **Implicit-Cert-Gen** algorithm:
   (a) The TA creates the explicit certificate that links S's identity with the public key components:
   $$eCert_{ID_s} = \frac{1}{s + r_{ID_s} q_{ID_s}} P \tag{3}$$
   (b) The TA sends $eCert_{ID_s}$ to an entity S.
6. **ShortTerm-Explicit-Cert-Gen** $(params, v, V_0, T_0, bstr, CI_{ID_s}, eCert_{ID_s}, \tau_{st})$: Taking as input any bitstring, the user's certificate information CI_{ID_s} and his/her long-term explicit certificate $eCert_{ID_s}$ (created for the period τ_{lt}) and a period τ_{st}, the TSA first checks if the user and his/her long-term explicit certificate are in the $RL_{TA,t}$. If that is so, the TSA rejects the update request. Otherwise, the TSA:
   (a) Randomly selects secret key $z \in Z_p^*$ and computes $(Z', Z'') = (zP, zQ)$;
   (b) Composes the certificate status information CSI_{ID_s}, including (Z', Z''), the TSA public keys (V_0, T_0), ID_s and ID_{TSA} identifiers, the status value equal to *good*, and the period τ_{st} for which the information CSI_{ID_s} should be valid;
   (c) For CI_{ID_s}, an explicit certificate $eCert_{ID_s}$ and CSI_{ID_s} computes:
   $$\begin{aligned} t_{ID_s} &= H_2(bstr, CI_{ID_s}, eCert_{ID_s}, CSI_{ID_s}) \\ I_{ID_s} &= (v + zt_{ID_s})(Q_0 + q_{ID_s} R''_{ID_s}) \end{aligned} \tag{4}$$
   where $q_{ID_s} = H_1(CI_{ID_s})$;
   (d) Generates the explicit short-term certificate (the certificate status evidence) as:
   $$e_{st}Cert_{ID_s} = \frac{1}{v + zt_{ID_s}} Q \tag{5}$$
   and transfers it to the user S via a public (open) channel; in addition, the TSA sends CSI_{ID_s} and I_{ID_s}.

7. **Set-Private-Key** $(params, CI_{ID_s}, CSI_{ID_s}, s_{ID_s}, iCert_{ID_s}, e_{st}Cert_{ID_s})$: The user S calculates the hash values q_{ID_s} and t_{ID_s} (see Equations (1) and (4)), and checks if $\hat{e}(q_{ID_s}R'_{ID_s} + P_0, iCert_{ID_s}) = \hat{e}(t_{ID_s}Z'_{ID_s} + V_0, e_{st}Cert_{ID_s}) = \hat{e}(P, Q) = g$; if in both cases the answer is positive, then the algorithm formulates a full private key in the form $Sk_{ID_s} = (s_{ID_s}, iCert_{ID_s}, e_{st}Cert_{ID_s})$.
8. **Set-Public-Key** $(params, CI_{ID_s})$: The user S with P_{ID_s}, R'_{ID_s} and R''_{ID_s} (taken from the user's certificate information CI_{ID_s}) sets his full long-term public key in the form $Pk_{ID_s} = (P_{ID_s}, R'_{ID_s}, R''_{ID_s})$. The TA publishes the resulting full long-term public key in its public repository and distributes it to all interested parties.
9. **Cert-Revoke** $(params, CI_{ID_s}, eCert_{ID_s})$: The user with CI_{ID_s} or any other authorized entity sends to TA a tuple $(CI_{ID_s}, eCert_{ID_s})$ with the explicit long-term certificate $eCert_{ID_s}$ to be revoked. After verifying the entity credentials to revoke the certificate, TA revokes it and places it on a signed revocation list $RL_{TA,t}$.
10. **Sign** $(params, m, CI_{ID_s}, Sk_{ID_s}, eCert_{ID_s})$: To sign a message $m \in \{0,1\}^*$, a signer S performs the following steps:
    (a) Picks two random numbers $k_1, k_2 \in_R Z_p^*$;
    (b) Computes the hash value $bstr = H_3(m, k_1P)$, and $q_{ID_s} = H_1(CI_{ID_s})$;
    (c) Generates a short-term explicit certificate by calling the ShortTerm-Explicit-Cert-Gen $(params, v, V_0, T_0, bstr, CI_{ID_s}, eCert_{ID_s}, \tau_{st}) \rightarrow (e_{st}Cert_{ID_s}, CSI_{ID_s}, I_{ID_s})$ function;
    (d) Generates the signature $\sigma = (h, w_1, w_2, E)$,

$$E = \frac{k_1 - k_2^{-1}h}{k_1h + s_{ID_s}}(iCert_{ID_s} + e_{st}Cert_{ID_s}) \tag{6}$$

    where $h = H_3(m, k_1P, U, q_{ID_s})$, $w_1 = k_1 - hs_{ID_s} \pmod p$, $w_2 = k_2(k_1h + s_{ID_s}) \pmod p$, while $U = e(P, T_0 + t_{ID_s}Z''_{ID_s} + Q_0 + q_{ID_s}R''_{ID_s})^{k_1k_2}$;
    (e) If in (6) $k_1h + s_{ID_s} = 0$, then repeat steps (a) and (b).

    Note. Each time a signature is generated, a fresh short-term explicit certificate is retrieved from the TSA (cf. ShortTerm-Explicit-Cert-Gen algorithm).
11. **Verify** $(params, m, \sigma, CI_{ID_s}, CSI_{ID_s}, I_{ID_s}, eCert_{ID_s}, e_{st}Cert_{ID_s})$: To verify the tuple containing the message, the signature and certificates, i.e., $(m, \sigma = (h, w_1, w_2, E), eCert_{ID_s}, e_{st}Cert_{ID_s}, I_{ID_s})$, V performs the following steps:
    (a) Computes q_{ID_s} (see Equation (1)) and then calculates values:

$$\begin{aligned} U' &= \hat{e}(\psi(I_{ID_s}), E)^{w_2}\hat{e}(eCert_{ID_s} + \psi(e_{st}Cert_{ID_s}), I_{ID_s})^h \\ \overline{k_1P} &= w_1P + hP_{ID_s} \end{aligned} \tag{7}$$

    (b) Computes $bstr = H_3(m, k_1P)$ and t_{ID_s} (see Equation (4));
    (c) If the status of the certificate $eCert_{ID_s}$ in the certificate status information CSI_{ID_s} is correct and (8) is valid, then returns *accept*, otherwise *reject*.

$$h \equiv H_3(m, \overline{k_1P}, U', q_{ID_s}) \tag{8}$$

Remark 2. *Note that during the indirect signature verification, the long- and short-term explicit certificates are validated ($eCert_{ID}$ and $etCert_{ID}$, respectively). This verification can also be performed directly based on the following formulas:*

$$\begin{aligned} g &\equiv \hat{e}(eCert_{ID_s}, q_{ID_s}R''_{ID_s} + Q_0) \\ g &\equiv \hat{e}\left(t_{ID_s}Z'_{ID_s} + V_0, e_{st}Cert_{ID_s}\right) \end{aligned} \tag{9}$$

If the conditions formulated in Equations (8)–(9) are met, it means that a signature is mathematically correct. It is the first postulate for a digital signature to be valid. The second one applies to

the validity of digital signatures at a semantical level that depends on the underlying validity model (Baier, H. et al. [1]).

Suppose we use a shell model and the verifier received the signature at time t_v called the verification time. Assuming that the TA's master private key and the TSA's master private status key are irrevocable signature keys, the semantic validity of the digital signature depends on a short- and long-term certificate validity (e_{st}Cert and eCert, respectively). Because both certificates are mathematically correct and (e_{st}Cert, eCert) certificates are issued with respective periods $\tau_{lt} = [\tau^i_{lt}, \tau^e_{lt}]$, $\tau_{st} = [\tau^i_{st}, \tau^e_{st}]$ and expiry dates τ^e_{lt}, τ^e_{st}, then a verifier checks if:

(a) *e_{st}Cert was certified by the TSA and the validity period τ_{st} of e_{st}Cert satisfies $\tau^i_{lt} \leq \tau^i_{st} < \tau^e_{st} \leq \tau^e_{lt}$;*

(b) *$t_v \in [\tau^i_{st}, \tau^e_{st}]$.*

When the above conditions are successful, the signature will be accepted as valid short-period non-repudiation evidence in whole period $\tau_{st} = [\tau^i_{st}, \tau^e_{st}]$.

Remark 3. *Based on the properties of the asymmetric bilinear map groups:*

$$eCert_{ID_s} = \psi(iCert_{ID_s}) \tag{10}$$

*Hence, it follows that, alternatively, the execution of the **LongTerm-Explicit-Cert-Gen** algorithm can be entrusted to the signatory S, who, after receiving the implicit certificate from the TA will use Equation (10) to calculate the explicit certificate.*

3.2. Correctness

The $\sigma = (h, w_1, w_2, E)$ is a valid signature on message m because it is accepted by **Verify**. We state the proof as follows:

$$\begin{aligned}
\boldsymbol{U'} &= \hat{e}(\psi(I_{ID_s}), E)^{w_2} \hat{e}(eCert_{ID_s} + \psi(e_{st}Cert_{ID_s}), I_{ID_s}))^h \\
&= \hat{e}\left(\psi(I_{ID_s})), \frac{k_1 - k_2^{-1}h}{k_1 h + s_{ID_s}}(iCert_{ID_s} + e_{st}Cert_{ID_s})\right)^{k_2(k_1 h + s_{ID_s})} \\
&\quad \hat{e}(eCert_{ID_s} + \psi(e_{st}Cert_{ID_s}), I_{ID_s})^h \\
&= \hat{e}(\psi(I_{ID_s}), (k_1 k_2 - h)(iCert_{ID_s} + e_{st}Cert_{ID_s})) \\
&\quad \hat{e}(\psi(I_{ID_s}), h(iCert_{ID_s} + e_{st}Cert_{ID_s})) \\
&= \hat{e}\left(\psi((v + zt_{ID_s})(Q_0 + q_{ID_s}R^{''}_{ID_s})), k_1 k_2 (\frac{1}{s + r_{ID_s} q_{ID_s}} + \frac{1}{v + zt_{ID_s}})Q\right) \\
&= \hat{e}(P, (s + q_{ID_s} r_{ID_s})Q + (v + zt_{ID_s})Q)^{k_1 k_2} \\
&= \hat{e}(P, Q_0 + q_{ID_s} R''_{ID_s} + T_0 + t_{ID_s} Z''_{ID_s})^{k_1 k_2} = \boldsymbol{U}
\end{aligned} \tag{11}$$

Thus,

$$\boldsymbol{h'} = H_3(m, \overline{k_1 P}, U', q_{ID_s}) = H_3(m, w_1 P + h P_{ID_s}, U', q_{ID_s}) = \boldsymbol{h} \tag{12}$$

Moreover, based on this, it is straightforward to prove the correctness of the long-term explicit certificate:

$$\begin{aligned}
\boldsymbol{g'} &= \hat{e}(eCert_{ID_s}, q_{ID_s} R''_{ID_s} + Q_0) = \hat{e}\left(\frac{1}{s + r_{ID_s} q_{ID_s}} P, (q_{ID_s} r_{ID_s} + s)Q\right) \\
&= \hat{e}(P, Q) = \boldsymbol{g}
\end{aligned} \tag{13}$$

and short-term explicit certificate:

$$\begin{aligned} g' &= \hat{e}\left(t_{ID_s} Z'_{ID_s} + V_0, e_{st} Cert_{ID_s},\right) = \hat{e}\left((t_{ID_s} z + v)P, \frac{1}{v + z t_{ID_s}} Q,\right) \\ &= \hat{e}(P, Q) = g \end{aligned} \tag{14}$$

4. Security Analysis

In Games I and II, the TSA is treated as an oracle that answers every query the challenger or adversary asks. It has been assumed that long-term certificates are not revoked in these two games. Therefore, all explicit short-term certificates issued by the TSA have the status *good*. In Game III, long-term certificates can be revoked. The TSA will not issue a short-term explicit certificate for the next validity period of τ_{st_i}. Because the adversary still owns the implicit and long-term explicit certificates, it can try to produce valid signatures even if the previous short-term explicit certificate is no longer valid. The adversary does not know its short explicit certificate for the new target period but can cooperate with legal users to obtain such a certificate.

We proved the IE-RCBS-kCAA scheme security by reducing the security of a higher-level construction to a lower-level primitive. In particular, we reduced the existence of an adversary by transforming the protocol into an algorithm that solves the corresponding k-mCAA problem or the discrete logarithm (DL) problem with non-negligible probability. To this end, we used a general forking lemma (Bellare and Neven [44]), similar to [30].

Table 1 below shows that in comparison with the A_1 and A_3 adversaries, the A_2 adversary's capabilities are greater (if only because he/she has access to the master private key and master private status key that belong to the TA and TSA, respectively). On the other hand, the capabilities of A_1 and A_3 adversaries are similar:

- A_3 knows the implicit certificates of users whose long-term explicit certificate has been revoked (in particular, it may be his/her certificate) but cannot obtain from the TSA any valid short-term explicit certificates related to them; the TSA will not respond to any request of the adversary to issue an explicit short-term certificate for the next period after the related long-term explicit certificate has been revoked; hence, the adversary, in order to forge the adversary's signature, must be able to calculate an explicit short-term certificate;
- A_1 does not know the implicit certificates of users who were indicated as targets of the adversary attack; however, since, in this case, none of the explicit long-term certificates were revoked, the TSA responds to every request to issue (also from the adversary) a short-term explicit certificate for the next validity period; hence, the adversary A_1 knows the explicit short-term certificates of all users, including those who are the targets of the attack, but must calculate the corresponding implicit certificates.

In both cases, after creating a valid forged signature, the adversaries A_1 and A_3 disclose the corresponding short- and long-term explicit certificate. It follows that challenger C with the help of adversary A_1 or A_3 could solve the computing k-mCAA problem. However, this is contrary to the assumption that the k-mCAA problem is a computationally difficult problem. Hence, the proposed IE-RCBS-kCAA signature scheme is provably secure against Types I and III adversaries, as demonstrated in Lemmas 1 and 3, respectively. In Lemma 2, we also prove that IE-RCBS-kCAA is secure against a Type II adversary.

Lemma 1. *Suppose the hash functions H_1, H_2 and H_3 are random oracles, and A_1 is a Type I adversary in Game I against the IE-RCBS-kCAA scheme. When the adversary A_1 has a non-negligible ϵ advantage over the IE-RCBS-kCAA scheme, then there is a reduction R_1 that solves the k-mCAA problem over the G_2 group with non-negligible probability:*

$$\varepsilon^{R_1}_{k-mCAA} \geq \frac{\varepsilon^2}{\gamma^2 e((q_I + q_E + q_S) + 1)^2} \tag{15}$$

where e is the base of the natural logarithm, q_I, q_E, cq_S and $\gamma = q_{H_3}$ are the upper bound on the number of queries sent to the respective Implicit-Cert-Gen-Query, LongTerm-Explicit-Cert-Gen-Query, Super-Sign-Query oracles and the H_3-Query oracle.

Proof. (sketch) According to the approach given in [30] (also compare Lemma 3), our reduction consists of two phases. First, we apply the intermediate algorithm B_1 (i.e., the wrapper) that interacts with adversary A_1 and it returns a side output. Second, we build a reduction algorithm R_1 that launches general forking algorithm F_{B_1} with wrapper B_1 that handles the simulation of the IE-RCBS-kCAA scheme environment to the actual adversary. The algorithm R_1 returns data that allow the correct solution of the k-mCAA problem to be obtained.

Assume that B_1 is given a random instance $\triangle = (G_2, p, P, sP, Q, sQ, (s + r_1q_1)^{-1}Q, \ldots, (s + r_kq_k)^{-1}Q)$ of the k-mCAA problem, where G_2 is a group with a large prime order p. For the master private key $s \in Z_p^*$ unknown to C and B_1, the goal is to compute $(r^*q^* + s)^{-1}Q$ for some $q^* \notin \{q_1, \ldots, q_k\}$, $r^*Q \notin \{r_1Q, \ldots, r_kQ\}$, and given $q_1, \ldots, q_k \in Z_p^*$, $r_1Q, \ldots, r_kQ$. In order to achieve this goal, we convert Type I adversary A_1 to algorithm B_1 (compare with Lemma 3). Finally, the reduction algorithm R_1 invokes a general forking algorithm F_{B_1} with the wrapper B_1 to solve the challenge $\triangle$.

Note that in comparison to Lemma 3, the simulation of *ShortTerm-Explicit-Cert-Gen-Query* is simpler because it is reasonable now to respond to each request of the adversary A_1 (no long-term certificate is revoked). What is more, this response is always provided by the TSA, which thus becomes a component of the simulation environment.

R_1 obtains two signature forgeries $\hat{\sigma}_i = (\hat{m}, \hat{h}_i, \hat{w_{1,i}}, \hat{w_{2,i}}, \hat{E}_i, P_{ID}, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$, $(i = 0, 1)$ for the message $\hat{m}$, partial public key P_{ID}, and long- and short-term explicit certificates $eCert_{ID}$, $e_{st}Cert_{ID}$. If both forgeries are valid, then R_1 obtains two sets of side outputs σ_0 and σ_1 where σ_i (for $i = 0, 1$) is written as $(\hat{\beta}_i, \hat{h}_i, \hat{f}_i, \hat{w_{2,i}}, \hat{c}_i, \hat{U}_i, \hat{E}_i, P_{ID}, \hat{C}_{ID}, C\hat{S}I_{ID}, \hat{R}'_{ID}, \hat{R}''_{ID}, C, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$. Moreover, we assume that $\hat{U}_0 = \hat{U}_1$, $q^* = \hat{c}_0 = \hat{c}_1$ and $R''_{ID} = r^*Q$ (compare Lemma 3). R_1 outputs *failure* and stops if both $\hat{\beta}_0$ and $\hat{\beta}_1$ are equal to 0.

Based on σ_0 and σ_1, the following equation is used:

$$\begin{aligned} &\hat{e}\big(\psi(I_{ID}), \hat{E}_0\big)^{\hat{w}_{2,0}} \hat{e}(eCert_{ID} + \psi(e_{st}Cert_{ID}), I_{ID})^{\hat{h}_0} = \\ &= \hat{e}\big(\psi(I_{ID}), \hat{E}_1\big)^{\hat{w}_{2,1}} \hat{e}(eCert_{ID} + \psi(e_{st}Cert_{ID}), I_{ID})^{\hat{h}_1} \end{aligned} \tag{16}$$

Equation (16) can be converted into:

$$\begin{aligned} &\hat{e}\Big(\psi(I_{ID}), \hat{w}_{2,0}\hat{E}_0 + \hat{h}_0(iCert_{ID} + e_{st}Cert_{ID})\Big) = \\ &= \hat{e}\Big(\psi(I_{ID}), \hat{w}_{2,1}\hat{E}_1 + \hat{h}_1(iCert_{ID} + e_{st}Cert_{ID})\Big) \end{aligned} \tag{17}$$

Eventually, the solution to the k-mCAA problem is:

$$iCert_{ID_s} = \frac{1}{(r^*q^* + s)}Q = \frac{(\hat{w}_{2,0}\hat{E}_0 - \hat{w}_{2,1}\hat{E}_1)}{(\hat{h}_0 - \hat{h}_1)} - e_{st}Cert_{ID} \tag{18}$$

where $q^* \notin \{q_1, \ldots, g_k\}$ and $r^*Q \notin \{r_1Q, \ldots, r_kQ\}$. Note that a public channel transmits all short-term explicit certificates. Hence, in particular, $e_{st}Cert_{ID_s}$ can be known both to adversary A_1 and algorithm R_1.

The success probability of a Super Type I adversary is calculated similarly to Lemma 3. We should consider the same four events $\neg E_1$, $\neg E_2$, $\neg E_3$ and $\neg E_1$ with one exception: the wrapper B_1 cannot fail during the simulation of the oracle *ShortTerm-Explicit-Cert-Gen-Query*. Finally, from the general forking lemma, the success probability $\varepsilon_{k-mCAA}^{R_1}$ can be expressed as in Equation (15).

This ends the sketch proof. □

Next, in Game II, applied to the Super Type II adversary where the adversary models the certified entity, we require that signers are honest and the TA registers their tuples $(ID, P_{ID}, eCert_{ID})$. The following lemma can be shown for this assumption with the use of a random oracle model:

Lemma 2. *Suppose the hash functions H_1, H_2 and H_3 are random oracles, and A_2 is a Type II adversary in Game II against the IE-RCBS-kCAA scheme. When the adversary A_2 has a non-negligible ϵ advantage over the IE-RCBS-kCAA scheme, there is a reduction R_2 that solves the DL problem over the G_1 group with non-negligible probability:*

$$\varepsilon_{k-mCAA}^{R_2} \geq \frac{\varepsilon^2}{\gamma e((q_R + q_C) + 1)^2} \tag{19}$$

where q_R, q_C and $\gamma = q_{H_2}$ are the upper bound on the number of respective queries sent to the Public-Key-Replacement-Query, Corruption-Query and H_2-Query oracles.

The proof is similar to the proof of [30] and is omitted here.

Lemma 3. *Suppose the hash functions H_1, H_2 and H_3 are random oracles, and A_3 is a Type III adversary in Game III against the IE-RCBS-kCAA scheme. When the adversary A_3 has a non-negligible ϵ advantage over the IE-RCBS-kCAA scheme, there is a reduction R_3 that solves the k-mCAA problem over the G_2 group with non-negligible probability:*

$$\varepsilon_{k-mCAA}^{R_3} \geq \frac{\varepsilon^2}{\gamma e((q_I + q_E + q_T + q_S) + 1)^2} \tag{20}$$

where q_I, q_E, q_T, q_S and $\gamma = q_{H_3}$ are the upper bound on the number of respective queries sent to the Implicit-Cert-Gen-Query, LongTerm-Explicit-Cert-Gen-Query, ShortTerm-Explicit-Cert-Gen-Query, Super-Sign-Query and H_3-Query oracles.

Proof. We begin by describing the B_3 wrapper and next demonstrate how R_3 reduction invokes the F_{B_3} algorithm on the B_3 wrapper to solve the *k*-mCAA problem. Suppose the adversary A_3 can make q_{H_1}, q_{H_2}, q_{H_3}, q_T and q_S queries to hash functions H_1, H_2, H_3, and the *ShortTerm-Explicit-Cert-Gen-Query* and *Super-Sign-Query* oracle.

Algorithm R_3 is given a random instance $\triangle = (G_1, G_2, p, P, sP, Q, sQ, (s + r_1q_1)^{-1}Q,$ $\ldots,$ $(s + r_kq_k)^{-1}Q)$ of the *k*-mCAA problem, where the master private key $s \in Z_p^*$ is unknown to *C* and B_3. The challenger *C* and direct algorithm R_3 are asked to calculate $(r^*q^* + s)^{-1}Q$ for some $q^* \notin \{q_1, \ldots, q_k\}$, $r^*Q \notin \{r_1Q, \ldots, r_kQ\}$, and given $q_1, \ldots, q_k \in Z_p^*$, $r_1Q, \ldots, r_kQ$.

Assume we are also given $t_1, \ldots, t_k \in Z_p^*$, $z_1Q, \ldots, z_kQ$, $z_1P, \ldots, z_kP$, $(v + z_1t_1)^{-1}Q,$ $\ldots,$ $(v + z_kt_k)^{-1}Q)$, $(v + z_1t_1)(sQ + q_1r_1Q)$, $\ldots$, $(v + z_kt_k)(sQ + q_kr_kQ)$. This allows us to simulate the *ShortTerm-Explicit-Cert-Gen-Query* behaviour for all unrevoked long-term implicit certificates.

1. The Wrapper

We demonstrate that Type III adversary A_3 can be converted to algorithm B_3 and then used to solve a random instance $\triangle$ of the *k*-mCAA problem. Assume that $\gamma = q_{H_3}$ and $H = Z_p$. Wrapper B_3 takes $\triangle$ as an argument with a set of random elements $q_1, \ldots, q_k \in Z_p^*$ and $h_1, \ldots, h_\gamma \in Z_p^*$ and returns a tuple (J, σ) where *J* refers to indices of the target H_3 query and where σ is the side output. B_3 maintains two counters *ctr* and *cin*, which are initially both set to one, and three lists L_{H_1}, L_{H_2} and L_{H_3} used to store the answers to the H_1, H_2 and H_3 random oracle queries. Wrapper B_3 interacts with adversary A_3 as follows (Algorithm 1).

Algorithm 1 $B_3(\triangle)$.

Initialize. $ctr = 1$, $cin = 1$, lists L_{H_1}, L_{H_2} and L_{H_3} are empty.
TA-Setup. B_3 sets P and Q as the generators of groups G_1 and G_2, respectively, sets TA's master public keys ($P_0 = sP$, $Q_0 = sQ$) and TSA's master public status keys ($V_0 = vP$, $T_0 = vQ$). We assume that master secret keys s and v are unknown to everyone, including B_3. Then, B_3 defines $params = \{G_1, G_2, G_T, p, \hat{e}, P, P_0, V_0, Q, Q_0, T_0, H_1, H_2, H_3\}$ and sends them to the adversary A_3.
Queries: A_3 can query the following oracles polynomial number of times.

1. *Create-User-Query* ($params$, ID). Let us assume that the query is about the identity of ID and that B_3 replies as described below:
   (a) B_3 scans list L_U with tuples in form $\langle ID_i, s_{ID_i}, Pk_{ID_i}\rangle$ to check whether $ID_i = ID$ and if it is true returns a previously defined value P_{ID_i}.
   (b) else, B_3 selects $s_{ID} \in_R Z_p$ at random and calculates public key $P_{ID} = s_{ID}P$. B_1 returns P_{ID} and stores the tuple $\langle ID_i, s_{ID}, P_{ID}\rangle$ in the L_U list.
2. H_1-*Query* (CI_{ID_i}). Algorithm B_3 maintains a list L_{H_1} of tuples $\langle CI_{ID_i}, P_{ID_i}, R^{'}_{ID_i}, R^{''}_{ID_i}, coin_i, cin_i, c_i, C_i, eCert_{ID_i}\rangle$. If B_3 or A_3 queries H_1, algorithm B_3 returns c_i directly when L_{H_1} contains a tuple $\langle CI_{ID_i}, P_{ID_i}, R^{'}_{ID_i}, R^{''}_{ID_i}, coin_i, cin_i, c_i, C_i, eCert_{ID_i}\rangle$. Else:
   (a) B_3 randomly selects $c \in_R Z_p^*$ and sets $coin = C = eCert_{ID_i} = \perp$ when the query is made explicitly by A_3 ($\perp$ denotes unknown fields to B_3);
   (b) else, B_3 flips a biased coin that outputs value $coin = 1$ with a probability of ς and $coin = 0$ with a probability of $1 - \varsigma$ (the ς will be optimized later); next:
      i. if $coin = 0$, B_3 selects cin^{th} ($1 \le cin \le k$) value $q_{cin} \in \{q_1, \ldots, q_k\}$ and sets $c = q_{cin}$, $C = (r_{cin}c + s)^{-1}Q$, $eCert_{ID_i} = \psi(C)$, $R^{''}_{ID_i} = r_{cin}Q$ and $R^{'}_{ID_i} = \psi(R^{''}_{ID_i})$;
      ii. else, if $coin_i = 1$, B_3 randomly selects $c \in_R Z_p^*$ and $R^{''}_{ID} \in G_2$ such that $c \notin \{q_1, \ldots, q_k\}$ and $R^{''}_{ID} \notin \{r_1Q, \ldots, r_kQ\}$, respectively; computes $R^{'}_{ID} = \psi(R^{''}_{ID})$ and sets $C = eCert_{ID_i} = \perp$;
   (c) $\langle CI_{ID_i}, P_{ID_i}, R^{'}_{ID_i}, R^{''}_{ID_i}, coin, cin, c, C, eCert_{ID_i}\rangle$ is stored in L_{H_1} and returns c as the answer.
3. H_2-*Query* ($CI_{ID_i}, eCert_{ID_i}, CSI_{ID_s}$). On receiving the H_2 query on ($CI_{ID_i}, eCert_{ID_i}, CSI_{ID_s}$), algorithm B_3 looks up the list L_{H_2}. If the corresponding entry already appears in L_{H_2} with a tuple $\langle CI_{ID_i}, eCert_{ID_i}, P_{ID_i}, CSI_{ID_s}, Z^{'}_{ID_i}, Z^{''}_{ID_i}, cin_i, coin_i, t_i, e_{st}Cert_{ID_i}, I_{ID_i}\rangle$, then B_3 responds with t_i. Otherwise:
   (a) if the query is made explicitly by A_3, B_3 randomly selects $t \in_R Z_p^*$ and sets $coin = 1$, $e_{st}Cert_{ID_i} = I_{ID_i} = \perp$;
   (b) otherwise, B_3 first call H_1-*Query* (CI_{ID_i}) oracle and as a result takes a tuple $\langle CI_{ID_i}, P_{ID_i}, R^{'}_{ID_i}, R^{''}_{ID_i}, coin_i, cin_i, c_i, C_i, eCert_{ID_i}\rangle$ form L_{H_1} list.
      i. if $coin_i = 0$, B_3 chooses $t_{cin_i} \in \{t_1, \ldots, t_k\}$ and sets $t = t_{cin_i}$, $e_{st}Cert_{ID_i} = (v + z_{cin}t)^{-1}Q$, $Z^{'}_{ID_i} = z_{cin}P$, $Z^{''}_{ID_i} = z_{cin}Q$, $I_{ID_i} = (v + z_{cin}t)(sQ + c_iR^{''}_{ID_i})$
      ii. otherwise, if $coin_i = 1$, B_3 randomly selects $t \in_R Z_p^*$ and $Z^{''}_{ID} \in G_2$ such that $t \notin \{t_1, \ldots, t_k\}$ and $Z^{''}_{ID} \notin \{z_1Q, \ldots, z_kQ\}$, respectively; computes $Z^{'}_{ID} = \psi(Z^{''}_{ID})$ and sets $e_{st}Cert_{ID_i} = I_{ID_i} = \perp$;
   (c) $\langle CI_{ID_i}, eCert_{ID_i}, P_{ID_i}, CSI_{ID_s}, Z^{'}_{ID_i}, Z^{''}_{ID_i}, cin_i, coin_i, t, e_{st}Cert_{ID_i}, I_{ID_i}\rangle$ is stored in L_{H_2} and t is output as the answer.
4. H_3-*Query* ($m, k_1P, U, H_1(CI_{ID_i})$). Algorithm B_3 maintains a list L_{H_3} of tuples $\langle m_i, (k_1P)_i, U_i, c_i, ctr, w_{2,i}, h_i\rangle$, where $c_i = H_1(CI_{ID_i})$. B_3 runs H_1-*Query* (CI_{ID_i}) and gets requested hash value c. For each request made on (m, k_1P, U, c), algorithm B_3 returns h_i directly when L_{H_3} contains tuple $\langle m_i, (k_1P)_i, U_i, c_i, ctr, w_{2,i}, h_i\rangle$. Else, B_3 returns $h = h_{ctr} \in_R Z_p^*$ as the output, adds tuple $\langle m_i, k_1P, U, c, ctr, \perp, h\rangle$ to L_{H_3} and increments ctr by one.

Algorithm 1 *Cont.*

5 *Public-Key-Replacement-Query* (ID, P_{ID}, P'_{ID}):

(a) B_3 tries to find a tuple $\langle ID_i, s_{ID_i}, P_{ID_i}\rangle$ in the L_U list such that $ID_i = ID$ and $P_{ID_i} = P_{ID}$. When this does not exist, B_3 outputs $\perp$.

(b) Else, B_3 replaces $\langle ID_i, s_{ID_i}, P_{ID_i}\rangle$ with $\langle ID_i, \perp, P'_{ID_i}\rangle$. In this case, the secret value associated with the new public key is not necessary to replace the public key.

6 *Corruption-Query (ID)*. B_3 browses the list L_U for ID and tries to find a tuple $\langle ID_i, s_{ID_i}, P_{ID_i}\rangle$, then returns $s_{ID_{si}}$ to A_3 when the user ID is registered. If this is not the case, B_3 selects a random number $s_{ID} \in_R Z_p$, sets $P_{ID} = s_{ID}P$, adds $\langle ID, s_{ID}, P_{ID}\rangle$ to the L_U list and returns s_{ID_i} to A_3.

7 *Implicit-Cert-Gen-Query* (ID, P_{ID}). At any moment, A_3 or B_3 can query this oracle based on identity ID and partial public key P_{ID}.

(a) On the running *Implicit-Cert-Gen-Query* for ID and P_{ID}, B_3 first checks list L_U. When a user with ID is not created, B_3 returns $\perp$.

(b) Now, B_3 tries to find tuple $\langle CI_{ID_i}, P_{ID_i}, R'_{ID_i}, R''_{ID_i}, coin_i, cin_i, c_i, C_i, eCert_{ID_i}\rangle$ in L_{H_1} that fulfils the following conditions: $CI_{ID_i}.ID \equiv ID$, $P_{ID_i} \equiv P_{ID}$ and $coin_i \neq \perp$. If such a tuple exists, then:

i. if $coin_i = 1$, *failure* (denoted by E_{11}) is returned and the simulation stops because it cannot respond to a query about any revoked user with an identity of ID and partial public key P_{ID};

ii. otherwise, B_3 outputs $(C_i, R'_{ID_i}, R''_{ID_i}, CI_{ID_i})$ to A_3 as the answer.

(c) Otherwise, B_3

i. creates CI_{ID} (see *Implicit-Cert-Gen* algorithm in Section 3.1), where $R'_{ID} = R''_{ID} = \perp$;

ii. runs H_1-*Query* (CI_{ID}) and repeats step (b).

8 *LongTerm-Explicit-Cert-Gen-Query* (CI_{ID}, P_{ID}). Upon receiving a query on identity ID with certificate information CI_{ID} and partial public key P_{ID}:

(a) B_3 first checks list L_U. If a user with $CI_{ID}.ID$ is not created, B_1 returns $\perp$.

(b) If there is a tuple $\langle CI_{ID_i}, P_{ID_i}, R'_{ID_i}, R''_{ID_i}, coin, cin_i, c_i, C_i, eCert_{ID_i}\rangle$ in L_{H_1} such that $CI_{ID_i} \equiv CI_{ID}$, $P_{ID_i} \equiv P_{ID}$ and $coin_i \neq \perp$, then:

i. if $coin_i = 1$, *failure* (denoted by E_{12}) is output and the simulation stops because it is not allowed to answer the query on any revoked user with the certificate information CI_{ID}, which is to be challenged;

ii. otherwise, it outputs $(eCert_{ID_i}, R'_{ID_i}, R''_{ID_i}, CI_{ID_i})$ to A_3 as the answer.

(c) Otherwise, B_3 runs H_1-*Query* (CI_{ID}) and repeats step (b).

9 *ShortTerm-Explicit-Cert-Gen-Query* $(CI_{ID}, P_{ID}, eCert_{ID})$.

(a) B_1 verifies list L_U. When a user with $CI_{ID}.ID$ does not exists, B_1 returns $\perp$.

(b) Now, B_3 checks in L_{H_2} list if there is a tuple $\langle CI_{ID_i}, eCert_{ID_i}, P_{ID_i}, CSI_{ID_s}, Z'_{ID_i}, Z''_{ID_i}, cin_i, coin_i, t, e_{st}Cert_{ID_i}, I_{ID_i}\rangle$ such that $CI_{ID} \equiv CI_{ID_i}$, $P_{ID} \equiv P_{ID_i}$, $eCert_{ID} \equiv eCert_{ID_i}$ and $coin_i \neq \perp$. Provided that such a tuple exists, then:

i. if $coin_i = 1$, *failure* (denoted by E_{13}) is returned the simulation stops since B_1 is not allowed to respond to the query on any revoked user with certificate information CI_{ID}, which is to be challenged;

ii. otherwise, B_3 outputs $(e_{st}Cert_i, CSI_{ID_i}, I_{ID_i})$;

(c) Otherwise, B_3

i. composes a certificate status information CSI_{ID} (see *ShortTerm-Explicit-Cert-Gen* algorithm in Section 3.1), where $Z'_{ID} = Z''_{ID} = \perp$;

ii. runs H_2-*Query* $(CI_{ID}, eCert_{ID}, CSI_{ID_s})$ and repeats step (b).

Algorithm 1 *Cont.*

10 *Super-Sign-Query* $(m, CI_{ID}, CSI_{ID}, P_{ID})$. B_3 returns $\perp$ if a user with $CI.ID$ does not exists. Else:

(a) if there is no a tuple $\langle CI_{ID_i}, P_{ID_i}, R'_{ID_i}, R''_{ID_i}, coin_i, cin_i, c_i, C_i, eCert_{ID_i}\rangle$ in L_{H_1} that fulfils the following conditions: $CI_{ID_i} \equiv CI_{ID}$, $P_{ID_i} \equiv P_{ID}$ and $coin_i \neq \perp$, B_3 runs H_1-*Query* (CI_{ID});

(b) if tuple $\langle CI_{ID_i}, eCert_{ID_i}, P_{ID_i}, CSI_{ID_s}, Z'_{ID_i}, Z''_{ID_i}, cin_i, coin_i, t, e_{st}Cert_{ID_i}, I_{ID_i}\rangle$ does not exists in L_{H_2} such that $CI_{ID} \equiv CI_{ID_i}$, $P_{ID} \equiv P_{ID_i}$, $CSI_{ID} \equiv CSI_{ID_i}$ and $coin_i \neq \perp$, B_3 runs H_2-*Query* $(CI_{ID}, eCert_{ID}, CSI_{ID_s})$;

(c) next, if $coin_i = 1$, B_3 reports *failure* (denoted by E_{14}) and terminates the simulation because it is not allowed to answer the sign query on any revoked and challenged user with certificate information CI_{ID};

(d) otherwise B_3 calculates the signature as follows:

i. sets $c = c_i$, $eCert_{ID} = eCert_{ID_i}$, $e_{st}Cert_{ID} = e_{st}Cert_{ID_i}$, $I_{ID} = I_{ID_i}$;

ii. selects $w_1, w_2 \in_R Z_p^*$ and $E \in_R G_1$ at random and sets $h = h_{ctr}$;

iii. calculates $U = \hat{e}(\psi(I_{ID}), E)^{w_2}\hat{e}(eCert_{ID} + \psi(e_{st}Cert_{ID}), I_{ID}))^h$ and $k_1P = w_1P + hP_{ID}$;

iv. B_3 tries to find tuple (m, k_1P, U, c) in the L_{H_3} list; if such a tuple appears in tuple $\langle m, (k_1P)_i, U_i, c_i, ctr_i, w_{w,i}, h_i\rangle$ of the L_{H_2} list, i.e., $m = m_i$, $k_1P = (k_1P)_i$, $U = U_i$ and $c = c_i$, B_3 increment index ctr by one and repeats from step (b) and point (ii);

v. B_3 adds tuple $\langle m, k_1P, U, c, ctr, w_2, h\rangle$ to L_{H_3} and increments ctr by one;

vi. B_3 returns tuple $(m, \sigma = (h, w_1, w_2, E), CI_{ID}, CSI_{ID}, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$ to A_3, where σ is the signature.

The sign query oracle does not use the user's secret value, which makes it a *Super-Sign* oracle.

Output. A successful adversary returns a valid forgery $(\hat{m}, \hat{\sigma} = (\hat{h}, \hat{w_1}, \hat{w_2}, \hat{E}), CI_{ID}, CSI_{ID}, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$ for $(\hat{m}, CI_{ID}, CSI_{ID}, P_{ID})$. Hence, we have $\hat{h} = H_3(\hat{m}, \hat{w_1}P + \hat{h}P_{ID}, U, \hat{c})$, where $U = \hat{e}(\psi(I_{ID}), \hat{E})^{\hat{w_2}}\ \hat{e}(eCert_{ID}+\psi(e_{st}Cert_{ID}), I_{ID})^{\hat{h}}$, $\hat{c} = H_1(CI_{ID})$ and $P_{ID} = \hat{s}_{ID}P$. In this instance, P_{ID} is chosen by A_3 and may not be the one returned by the oracle *Create-User-Query*. Moreover, $(CI_{ID}, P_{ID}, eCert_{ID})$ and $(\hat{m}, CI_{ID}, CSI_{ID}, P_{ID})$ have never appeared as *ShortTerm-Explicit-Cert-Gen-Query* or *Super-Sign-Query* queries, respectively.

Let $\langle CI_{ID}, P_{ID}, R'_{ID}, R''_{ID}, coin, cin, \hat{c}, C, eCert_{ID}\rangle$, $\langle CI_{ID}, eCert_{ID}, P_{ID}, CSI_{ID}, Z'_{ID}, Z''_{ID}, cin, coin, \hat{t}, \perp, \perp\rangle$ and $\langle \hat{m}, \hat{w_1}P + \hat{h}P_{ID}, U, \hat{c}, ctr_i, \hat{w_2}, \hat{h}\rangle$ be the respective tuples of L_{H_1}, L_{H_2} and L_{H_3} that correspond to the target valid forgery $\hat{\sigma}$. Thus, wrapper B_3 returns $(ctr_i, coin, \hat{h}, \hat{t}, \hat{w_2}, \hat{c}, U, \hat{E}, P_{ID}, \hat{CI}_{ID}, \hat{CSI}_{ID}, \hat{R}'_{ID}, \hat{R}''_{ID}, C, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$ as its output.

Note that side output σ consists of $(coin, \hat{h}, \hat{t}, \hat{w_2}, \hat{c}, U, \hat{E}, P_{ID}, \hat{CI}_{ID}, \hat{CSI}_{ID}, \hat{R}'_{ID}, \hat{R}''_{ID}, C, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$. In order to achieve these side output components, we assume that tuple $(\hat{m}, \hat{w_1}P + \hat{h}P_{ID}, U, \hat{c})$ has been queried to random oracle H_3-*Query* and the tuple $\langle \hat{m}, \hat{w_1}P + \hat{h}P_{ID}, U, \hat{c}, ctr_i, \hat{w_2}, \hat{h}\rangle$ is given in L_{H_3} list).

When an adversary returns an *invalid* forgery, B_3 returns *failure* (denoted by E_2) and aborts.

2. Reduction Algorithm R_3

Now we can show how to build a reduction algorithm R_3 that can exploit the general forking algorithm related with the above wrapper B_3. Let $\triangle = (G_1, G_2, p, P, sP, Q, sQ, (s + r_1q_1)^{-1}Q, \ldots, (s + r_kq_k)^{-1}Q)$ be the given k-mCAA problem. Reducing Algorithm R_3 invokes the general forking algorithm F_{B_3} to solve the k-mCAA problem (Algorithm 2).

Algorithm 2 $R_3(\triangle)$.

$(b, \{\sigma_0, \sigma_1\}) \xleftarrow{\$} F_{B_3}(\triangle)$
`if (b == 0) then return 0 // Event E3 (`F_{B_3}` fails and stops)`
`parse` σ_i `as`
$(\hat{\beta}_i, \hat{h}_i, \hat{f}_i, \hat{w_{2,i}}, \hat{c}_i, \hat{U}_i, \hat{E}_i, P_{ID}, \hat{CI}_{ID}, \hat{CSI}_{ID},$
$\hat{R}'_{ID}, \hat{R}''_{ID}, C, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$
`let` $\hat{\beta} = \hat{\beta}_0 = \hat{\beta}_1, \hat{U}_0 = \hat{U}_1, q^*{=}\hat{c}_0 = \hat{c}_1$ `and` $\hat{t}_0 = \hat{t}_1$
`if` $\hat{\beta} == 1$ `then`
`return` $iCert_{ID} = (\hat{h}_0 - \hat{h}_1)^{-1}(\hat{w_{2,0}}\hat{E}_0 - \hat{w_{2,1}}\hat{E}_1) - e_{st}Cert_{ID}$
`else return 0 // Event E4 (`F_{B_3}` is successful)`

3. Correctness of the k-mCAA Problem Solution

When the general forking algorithm F_{B_3} does not fail, R_3 obtains two sets of side outputs σ_0 and σ_1, where σ_i (for $i = 0, 1$) is written as $(\hat{\beta}_i, \hat{h}_i, \hat{f}_i, \hat{w_{2,i}}, \hat{c}_i, \hat{U}_i, \hat{E}_i, P_{ID}, \hat{C}_{ID}, \hat{CSI}_{ID}, \hat{R}'_{ID}, \hat{R}''_{ID}, C, eCert_{ID}, e_{st}Cert_{ID}, I_{ID})$. Compare output σ from wrapper B_3 to measure this phenomenon. Additionally, we assume that $\hat{\beta} = \hat{\beta}_0 = \hat{\beta}_1, \hat{U}_0 = \hat{U}_1, q^* = \hat{c}_0 = \hat{c}_1, \hat{t}_0 = \hat{t}_1$ and $R''_{ID} = r^*Q$. R_3 returns *failure* (denoted by E_4) and stops if $\hat{\beta}$ is equal to 0.

Algorithm R_3 obtains two valid signature forgeries $\hat{\sigma}_i = (\hat{m}, \hat{\sigma} = (\hat{h_i}, \hat{w_{1,i}}, \hat{w_{2,i}}, \hat{E}_i), CI, CSI, eCert, e_{st}Cert_{ID}, I_{ID})$ $(i = 0, 1)$ for the same message $\hat{m}$, public key P_{ID}, long-term explicit certificate $eCert_{ID}$ and short-term explicit certificate $e_{st}Cert_{ID}$. The following equation is applied based on two sets of side outputs σ_0 and σ_1:

$$\begin{aligned} &\hat{e}\big(\psi(I_{ID}), \hat{E}_0\big)^{\hat{w}_{2,0}} \hat{e}(eCert_{ID} + \psi(e_{st}Cert_{ID}), I_{ID})^{\hat{h}_0} = \\ &= \hat{e}\big(\psi(I_{ID}), \hat{E}_1\big)^{\hat{w}_{2,1}} \hat{e}(eCert_{ID} + \psi(e_{st}Cert_{ID}), I_{ID})^{\hat{h}_1} \end{aligned} \tag{21}$$

By making suitable arrangements:

$$\begin{aligned} &\hat{e}\Big(\psi(I_{ID}), \hat{w}_{2,0}\hat{E}_0 + \hat{h}_0(iCert_{ID} + e_{st}Cert_{ID})\Big) = \\ &= \hat{e}\Big(\psi(I_{ID}), \hat{w}_{2,1}\hat{E}_1 + \hat{h}_1(iCert_{ID} + e_{st}Cert_{ID})\Big) \end{aligned} \tag{22}$$

Eventually, the k-mCAA problem solution is:

$$iCert_{ID} = \frac{1}{(r^*q^* + s)}Q = \frac{(\hat{w}_{2,0}\hat{E}_0 - \hat{w}_{2,1}\hat{E}_1)}{(\hat{h}_0 - \hat{h}_1)} - e_{st}Cert_{ID} \tag{23}$$

where $q^* \notin \{q_1, \ldots, q_k\}$ and $\hat{r^*}Q \notin \{r_1Q, \ldots, r_kQ\}$.

The probability that the R_3 algorithm will solve the k-mCAA problem has not yet been calculated. In accordance with the simulation results, the R_3 algorithm can compute the value of $iCert_{ID}$ if and only if the below events occur:

- $\neg E_1$: B_3 does not fail during the simulation;
- $\neg E_2$: A_3 outputs a valid forgery;
- $\neg E_3$: F_{B_3} does not fail;
- $\neg E_4$: R_3 does not fail, i.e., in interaction with adversary A_3 outputs two valid forgeries with a coin value $\hat{\beta}$ of 1.

We denote the probability with which F_{B_3} succeeds during the first run as acc_3. Since F_{B_3} succeeds during the first run when there is no interruption in the query phase (event E_1 does not occur) and when adversary A_3 creates a valid forgery (event E_2 does not occur), we have:

$$acc_3 \geq Pr[\neg E_1 \wedge \neg E_2] = Pr[\neg E_1]Pr[\neg E_2|\neg E_1] \tag{24}$$

Event $\neg E_1$ occurs only when the four following events for $\neg E_1$ happen:

$\neg E_{11}$: B_3 cannot terminate during oracle simulation *Implicit-Cert-Gen-Query*, which occurs with a probability of $(1-\zeta)^{q_I}$;

$\neg E_{12}$: B_3 cannot terminate during oracle simulation *LongTerm-Explicit-Cert-Gen-Query*, which occurs with a probability of $(1-\zeta)^{q_E}$;

$\neg E_{13}$: B_3 cannot terminate during oracle simulation *ShortTerm-Explicit-Cert-Gen-Query*, which occurs with a probability of $(1-\zeta)^{q_T}$;

$\neg E_{14}$: B_3 cannot terminate during oracle simulation *Super-Sign-Query*, which occurs with a probability of $(1-\zeta)^{q_S}$.

Then we obtain:

$$Pr[\neg E_1] = Pr[\neg E_{11} \wedge \neg E_{12} \wedge \neg E_{13} \wedge \neg E_{14}] = (1-\zeta)^{\mu} \tag{25}$$

where $\mu = q_I + q_E + q_T + q_S$.

In addition, the probability of adversary A_3 producing a valid forgery when event E_1 does not occur is equal to $Pr[\neg E_2 | \neg E_1] = \varepsilon$.

If events E_3 and E_4 do not occur, then the advantage of the algorithm R_3 in solving the k-mCAA problem is:

$$Pr[\neg E_3 \wedge \neg E_4] = Pr[\neg E_3] Pr[\neg E_4 | \neg E_3] \tag{26}$$

Let $gfrk$ be the probability at which F_{B_3} is successful. As event E_4 occurs when F_{B_3} fails:

$$Pr[\neg E_3] = gfrk \tag{27}$$

Based on the general forking lemma [30,44] for $\gamma = q_{H_3}$ and $|H| = p$:

$$gfrk \geq acc_3 \left(\frac{acc_3}{\gamma} - \frac{1}{p} \right) \geq (1-\zeta)^{\mu} \varepsilon \left(\frac{(1-\zeta)^{\mu} \varepsilon}{\gamma} - \frac{1}{p} \right) \tag{28}$$

The probability at which the event E_4 does not occur, when the event E_3 does not occur, is equal to the probability at which the coin's value of $\hat{\beta}$ valid forgeries is not equal to 0. Hence:

$$Pr[\neg E_4 | \neg E_3] = \zeta^2 \tag{29}$$

Finally, the derivative R_3 of a successful solution to the k-mCAA problem is computed as described below:

$$\varepsilon^{R_3}_{k-mCAA} \geq \zeta^2 (1-\zeta)^{\mu} \varepsilon \left(\frac{(1-\zeta)^{\mu} \varepsilon}{\gamma} - \frac{1}{p} \right) \tag{30}$$

When $p >> 1$, the expression is maximised at $\zeta = 1/\mu$. Therefore:

$$\begin{aligned} \varepsilon^{R_3}_{k-mCAA} &\geq \zeta^2 \frac{(1-\zeta)^{2\mu} \varepsilon^2}{\gamma} = \\ &= \frac{1}{(\mu+1)^2} \left(1 - \frac{1}{\mu+1} \right)^{2\mu} \frac{\varepsilon^2}{\gamma} \geq \\ &\geq \frac{\varepsilon^2}{\gamma e (\mu+1)^2} = \frac{\varepsilon^2}{\gamma e ((q_I + q_E + q_T + q_S) + 1)^2} \end{aligned} \tag{31}$$

□

5. Performance Analysis

We compared our proposed IE-RCBS-kCAA scheme in terms of performance to two other schemes of similar design. For computational comparisons, we use notations of time-consuming operations based on results presented in Y. Huang et al. [39]:

- T_p: the time of executing a bilinear pairing operation $\hat{e} : G_1 \times G_2 \rightarrow G_T$;
- T_m: the time of executing a scalar multiplication in G_1 and G_2;
- T_e: the time of an exponentiation operation in G_T;
- T_h: the time of executing a map-to-point hash function.

Many operations are several orders of magnitude faster than bilinear pairing (T_p), map-to-point hash (T_h), scalar multiplication (T_m) in G_1 or G_2 and exponentiation in G_T (T_e). These operations are hashing, inversion in Z_p^*, multiplication in Z_p^*, addition in Z_p^*, multiplication in G_T, and addition with G_1 or G_2. Therefore, we only consider T_p, T_m, T_e and T_h when we compare the computation costs of analysed signature schemes.

Table 2 includes a comparison of the IE-RCBS-kCAA scheme to two other schemes ($|G_1|$, $|G_2|$ and $|Z_p|$ mean the bit lengths of the elements in G_1, G_2 and Z_p, respectively). Our scheme has the same level of security as the two other schemes. Furthermore, our scheme IE-RCBS-kCAA contains a comparable number of time-consuming operations. In comparison with other schemes, the most time-consuming algorithm in IE-RCBS-kCAA is the *Sign* algorithm. However, the signature verification *Verify* algorithm is less time-consuming and requires only two bilinear pairing calculations instead of four. This feature of our algorithm is significant because, in practice, the signature operation is performed once, and the verification operation can be performed many times. However, obtaining this property resulted in our scheme's larger signature size.

Table 2. Comparisons between our scheme and two other schemes.

Scheme	Type	Signature Size	Sign	Verify	Security Level
SZS Y. Sun et al. [24]	CLS	$2\|G_1\|$	$3T_m + 2T_h$	$4T_p + 3T_h$	Super for A_1, A_2, A_3
HTH Y. Huang et al. [39]	CLS	$\|G_1\|$	$2T_m + 2T_h$	$4T_p + T_m + 3T_h$	Super for A_1, A_2, A_3
Proposed IE-RCBS-kCAA	IE-CBS	$\|G_2\| + 3\|Z_p\|$	$T_p + 5T_m$	$2T_p + 4T_m$	Super for A_1, A_2, A_3

The run times of the *Sign* and *Verify* algorithms were examined using a test computer (Intel Core i7-8750H@2,20 GHz, 16 GB RAM) with a single thread. The scheme was implemented using an IAIK ECCelerate library for Java. Type 2 asymmetric pairing (ate pairing) based on the Barreto–Naehrig curve was used with different field sizes (160, 256, 384, 512 and 638 bits). The results (Table 3) average five repetitions. The time required to execute the *Sign* and *Verify* algorithms is less than 100 ms (except for the filed size of 638 bits). In Table 3, we also give the size in bytes of the signature for the different field sizes. We obtained this value by serialising all signature elements into an array of bytes (the signature consists of one element in G_2 and 3 in Z_p). The resulting signature size ranges from 387 to 1250 bytes. In real applications, adding a few dozen more bytes for formatting and metadata would be necessary.

Table 3. Sign and Verify execution time, signature size.

Field Size (bits)	Sign (ms)	Verify (ms)	Signature Size (bytes)
160	20	12	387
256	28	23	560
384	50	45	789
512	79	70	1020
638	138	175	1250

6. Conclusions

In the IE-RCBS-kCAA scheme, a trusted status authority (TSA) directly performs the revocation procedure and, as a result, alleviates the load of the trusted authority (TA). Moreover, the architecture framework proposed in this paper can contain many TSA authorities, effectively solving the scalability problem. This property is unique and is not available in other similar solutions. Unlike other proposals, e.g., Jia et al. [27], obtaining such a solution is possible because a master private status key can be generated independently by each TSA.

We also formalize the security model for an adversary that can obtain a valid signature in the super sign query phase relevant to the public key selected by itself without providing the corresponding secret key. This security model proves that our IE-RCBS-kCAA signature scheme is semantically secure against Super Type I/II/III attackers under the k-mCAA and the DL assumptions over the respective groups G_2 and G_1.

The digital signature schemes must include a revocation mechanism to support non-repudiation and achieve Girault trust level 3. In order to achieve such properties, the paper proposes an architecture framework adapted to the needs of the IE-RCBS-kCAA scheme. In this scheme, an important role from the point of view of practical applications is played by the trusted status authority (TSA, see Figure 1), which periodically updates the explicit short-term certificates as long as the long-term explicit certificate has not been revoked. It is worth noting that an architecture framework (see Figure 1) can contain many TSA authorities, which effectively solves the scalability problem. This feature is very important when the TSA is applied to updating the validity of many short-term explicit certificates in the same or different trust domains.

The performance comparisons with two other revocable signature schemes show that our signature scheme has a similar number of time-intensive operations. The empirical execution times of the *Sign* and *Verify* algorithms are significantly below one second. The signature size in our scheme is approximately 1 KB. Such a size will be too large a value for some practical applications, e.g., the transmission of short messages from IoT sensors. However, this value will be within an acceptable range for large documents and for signing transactions that are sent over high-speed internet connections. The performance analysis shows that the scheme is suitable for further practical work when signing large documents is needed. In this case, the signature is always decidedly shorter than the message. In other practical applications, one must carefully analyse if the signature size is acceptable.

Future research includes deploying the signature scheme with the certificates to be validated according to the chain model (see Section 1). Such a signature scheme extends the feature of the IE-RCBS-kCAA scheme and should provide a non-repudiation property over a long period. Another important research problem will concern potential inconsistencies between the proposed mathematical proof of security of the encryption scheme and its real implementation. To compare different security models and evaluate their impact on the encrypted scheme, we plan to use a ranking based on pairwise comparisons (first proposed in Janicki et al. [45]). This approach will allow us to locate inconsistencies in the adopted security models and improve the design process of the encryption scheme.

Author Contributions: Conceptualization, J.P., T.H. and W.Z.; methodology, J.P.; software, T.H.; validation, J.P., T.H. and W.Z.; formal analysis, J.P. and T.H.; writing—original draft preparation, J.P., T.H. and W.Z.; writing—review and editing, J.P., T.H. and W.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baier, H.; Karatsiolis, V. Validity Models of Electronic Signatures and Their Enforcement in Practice. In Proceedings of the Public Key Infrastructures, Services and Applications: 6th European Workshop, EuroPKI 2009, Pisa, Italy, 10–11 September 2009; Revised Selected Papers; Martinelli, F., Preneel, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 255–270. [CrossRef]
2. Ben, M.; Barka, M.; Krief, F.; Ly, O. Modeling Long-Term Signature Validation for Resolution of Dispute. In Proceedings of the Theory of Security and Applications: Joint Workshop, TOSCA 2011, Saarbrücken, Germany, 31 March–1 April 2011; Revised Selected Papers; Mödersheim, S.; Palamidessi, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 78–97. [CrossRef]
3. Hyla, T.; Pejaś, J. A Hess-like Signature Scheme based on Implicit and Explicit Certificates. *Comput. J.* **2017**, *60*, 457–475. [CrossRef]
4. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In Proceedings of the Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, CA, USA, 19–22 August 1984; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1984; Volume 196, pp. 47–53. [CrossRef]
5. Desmedt, Y.; Burmester, M. Identity-Based Key Infrastructures (IKI). In Proceedings of the Security and Protection in Information Processing Systems: IFIP 18th World Computer Congress TC11 19th International Information Security Conference, Toulouse, France, 22–27 August 2004; Deswarte, Y., Cuppens, F., Jajodia, S., Wang, L., Eds.; Springer US: Boston, MA, USA, 2004; pp. 167–176. [CrossRef]
6. Hyla, T.; Pejaś, J. Non-standard Certification Models for Pairing Based Cryptography. In Proceedings of the Hard and Soft Computing for Artificial Intelligence, Multimedia and Security, Miedzyzdroje, Poland, 19–21 October 2016 ; Kobayashi, S.Y., Piegat, A., Pejaś, J., El Fray, I., Kacprzyk, J., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 167–181. [CrossRef]
7. Girault, M. Self-certified public keys. In Proceedings of the Advances in Cryptology—EUROCRYPT '91: Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, 8–11 April 1991; Davies, D.W., Ed.; Springer: Berlin/Heidelberg, Germany, 1991; pp. 490–497. [CrossRef]
8. Ju, H.S.; Kim, D.Y.; Lee, D.H.; Lim, J.; Chun, K. Efficient Revocation of Security Capability in Certificateless Public Key Cryptography. In Proceedings of the Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference, KES 2005, Melbourne, Australia, 14–16 September 2005; Part II; Khosla, R., Howlett, R.J., Jain, L.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 453–459. [CrossRef]
9. Chow, S.S.M.; Boyd, C.; Nieto, J.M.G. Security-Mediated Certificateless Cryptography. In Proceedings of the Public Key Cryptography - PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, 24–26 April 2006; Yung, M., Dodis, Y., Kiayias, A., Malkin, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 508–524. [CrossRef]
10. Wu, T.Y.; Tsai, T.T.; Tseng, Y.M. A Provably Secure Revocable ID-Based Authenticated Group Key Exchange Protocol with Identifying Malicious Participants. *Sci. World J.* **2014**, *2014*, 10. [CrossRef]
11. Al-Riyami, S.S. Cryptographic Schemes Based on Elliptic Curve Pairings. Ph.D. Thesis, Information Security Group, Department of Mathematics Royal Holloway, University of London, London, UK, 2004.
12. Abinav, K.; Badrinarayanan, S.; Rangan, C.P.; Selvi, S.S.D.; Vivek, S.S.; Pradhan, V.K. A Revocable Online-Offline Certificateless Signature Scheme without Pairing. *IACR Cryptol. Eprint Arch., Paper 2013/758* **2013**, *2013*. Available online: https://eprint.iacr.org/2013/758 (accessed on 20 June 2023).
13. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In Proceedings of the Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; Kilian, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 213–229. [CrossRef]
14. Boldyreva, A.; Goyal, V.; Kumar, V. Identity-based Encryption with Efficient Revocation. In Proceedings of the 15th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 27–31 October 2008; CCS '08; ACM: New York, NY, USA, 2008; pp. 417–426. [CrossRef]
15. Libert, B.; Vergnaud, D. Adaptive-ID Secure Revocable Identity-Based Encryption. In Proceedings of the Topics in Cryptology—CT-RSA 2009: The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, 20–24 April 2009; Fischlin, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–15. [CrossRef]
16. Seo, J.H.; Emura, K. Revocable Identity-Based Encryption Revisited: Security Model and Construction. In Proceedings of the Public-Key Cryptography—PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, 26 February–1 March 2013; Kurosawa, K., Hanaoka, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 216–234. [CrossRef]
17. Wu, T.Y.; Tsai, T.T.; Tseng, Y.M. Revocable ID-based Signature Scheme with Batch Verifications. In Proceedings of the 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Piraeus-Athens, Greece, 18–20 July 2012; pp. 49–54. [CrossRef]
18. Tseng, Y.M.; Tsai, T.T. Efficient Revocable ID-Based Encryption with a Public Channel. *Comput. J.* **2012**, *55*, 475–486. [CrossRef]
19. Wu, T.Y.; Lin, J.C.W.; Chen, C.M.; Tseng, Y.M.; Frnda, J.; Sevcik, L.; Voznak, M. A brief review of revocable ID-based public key cryptosystem. *Perspect. Sci.* **2016**, *7*, 81–86. [CrossRef]
20. Chen, J.; Lim, H.W.; Ling, S.; Wang, H.; Nguyen, K. Revocable Identity-Based Encryption from Lattices. In Proceedings of the Information Security and Privacy: 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, 9–11 July 2012; Susilo, W., Mu, Y., Seberry, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 390–403. [CrossRef]

21. Cheng, S.; Zhang, J. Adaptive-ID Secure Revocable Identity-Based Encryption from Lattices via Subset Difference Method. In Proceedings of the Information Security Practice and Experience: 11th International Conference, ISPEC 2015, Beijing, China, 5–8 May 2015; Lopez, J., Wu, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 283–297. [CrossRef]
22. Lee, K.; Lee, D.H.; Park, J.H. Efficient revocable identity-based encryption via subset difference methods. *Des. Codes Cryptogr.* **2017**, *85*, 39–76. [CrossRef]
23. Lee, K.; Park, J.H. Identity-Based Revocation From Subset Difference Methods Under Simple Assumptions. *IEEE Access* **2019**, *7*, 60333–60347. [CrossRef]
24. Sun, Y.; Zhang, F.; Shen, L. A Revocable Certificateless Signature Scheme. *J. Comput.* **2014**, *9*, 1843–1850. [CrossRef]
25. Sun, Y.; Shen, L. Pairing-Free and Revocable Certificateless Signature Against Signing Key Exposure. *J. Emerg. Trends Comput. Inf. Sci.* **2014**, *5*, 845–849.
26. Sun, Y.; Zhang, Z.; Shen, L. A Revocable Certificateless Signature Scheme Without Pairing. In Proceedings of the Cloud Computing and Security: Second International Conference, ICCCS 2016, Nanjing, China, 29–31 July 2016; Revised Selected Papers, Part I; Sun, X., Liu, A., Chao, H.C., Bertino, E., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 355–364. [CrossRef]
27. Jia, X.; He, D.; Zeadally, S.; Li, L. Efficient Revocable ID-Based Signature With Cloud Revocation Server. *IEEE Access* **2017**, *5*, 2945–2954. [CrossRef]
28. Ma, M.; Shi, G.; Shi, X.; Su, M.; Li, F. Revocable Certificateless Public Key Encryption With Outsourced Semi-Trusted Cloud Revocation Agent. *IEEE Access* **2020**, *8*, 148157–148168. [CrossRef]
29. Yum, D.H.; Lee, P.J. Separable Implicit Certificate Revocation. In Proceedings of the Information Security and Cryptology—ICISC 2004: 7th International Conference, Seoul, Korea, 2–3 December 2004; Revised Selected Papers; Park, C.S., Chee, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 121–136. [CrossRef]
30. Hyla, T.; Pejaś, J. Demonstrably Secure Signature Scheme Resistant to *k*-Traitor Collusion Attack. *IEEE Access* **2018**, *6*, 50154–50168. [CrossRef]
31. Mitsunari, S.; Sakai, R.; Kasahara, M. A New Traitor Tracing. *IEICE Trans. A* **2002**, *85*, 481–484.
32. Santesson, S.; Myers, M.; Ankney, R.; Malpani, A.; Galperin, S.; Adams, D.C. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol–OCSP. *pkix* **2013**, RFC 6960. [CrossRef]
33. Hyla, T.; Pejaś, J. A Signature Scheme Based on Implicit and Explicit Certificates Against k-Traitors Collusion Attack. In Proceedings of the Computer Information Systems and Industrial Management, Bialystok, Poland, 16–18 June 2017; Saeed, K., Homenda, W., Chaki, R., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 638–651.
34. Viswanadham, Y.V.R.S.; Jayavel, K. A Framework for Data Privacy Preserving in Supply Chain Management Using Hybrid Meta-Heuristic Algorithm with Ethereum Blockchain Technology. *Electronics* **2023**, *12*, 1404. [CrossRef]
35. Koczkodaj, W.; Mansournia, M.; Pedrycz, W.; Wolny-Dominiak, A.; Zabrodskii, P.; Strzałka, D.; Armstrong, T.; Zolfaghari, A.; Dębski, M.; Mazurek, J. 1,000,000 cases of COVID-19 outside of China: The date predicted by a simple heuristic. *Glob. Epidemiol.* **2020**, *2*, 100023. [CrossRef]
36. Craven, M.J.; Woodward, J.R. Evolution of group-theoretic cryptology attacks using hyper-heuristics. *J. Math. Cryptol.* **2022**, *16*, 49–63. [CrossRef]
37. Koczkodaj, W.W. Statistically Accurate Evidence of Improved Error Rate by Pairwise Comparisons. *Percept. Mot. Ski.* **1996**, *82*, 43–48. [CrossRef]
38. Bellare, M.; Rogaway, P. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the ACM Conference on Computer and Communications Security, Fairfax, VA, USA, 3–5 November 1993; pp. 62–73.
39. Hung, Y.; Tseng, Y.; Huang, S. A revocable certificateless short signature scheme and its authentication application. *Informatica* **2016**, *27*, 549–572. [CrossRef]
40. Li, J.; Huang, X.; Mu, Y.; Susilo, W.; Wu, Q. Certificate-Based Signature: Security Model and Efficient Construction. In Proceedings of the Public Key Infrastructure: 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, 28–30 June 2007; Lopez, J., Samarati, P., Ferrer, J.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 110–125. [CrossRef]
41. Li, J.; Huang, X.; Zhang, Y.; Xu, L. An efficient short certificate-based signature scheme. *J. Syst. Softw.* **2012**, *85*, 314–322. [CrossRef]
42. Li, J.; Huang, X.; Mu, Y.; Susilo, W.; Wu, Q. Constructions of certificate-based signature secure against key replacement attacks. *J. Comput. Secur.* **2010**, *18*, 421–449. [CrossRef]
43. Huang, X.; Mu, Y.; Susilo, W.; Wong, D.S.; Wu, W. Certificateless Signatures: New Schemes and Security Models. *Comput. J.* **2012**, *55*, 457–474. [CrossRef]
44. Bellare, M.; Neven, G. Multi-signatures in the Plain public-Key Model and a General Forking Lemma. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Copenhagen, Denmark, 26–30 November 2023; CCS '06; ACM: New York, NY, USA, 2006; pp. 390–399. [CrossRef]
45. Janicki, R.; Koczkodaj, W. A weak order approach to group ranking. *Comput. Math. Appl.* **1996**, *32*, 51–59. [CrossRef]