Research Article

Vishal Saraswat*, Rajeev Anand Sahu and Amit K. Awasthi

A secure anonymous proxy signcryption scheme

DOI: 10.1515/jmc-2015-0014
Received March 12, 2015; revised February 2, 2017; accepted March 21, 2017

Abstract: We introduce a new cryptographic primitive identity-based anonymous proxy signcryption which provides anonymity to the proxy sender while also providing a mechanism to the original sender to expose the identity of the proxy sender in case of misuse. We introduce a formal definition of an identity-based anonymous proxy signcryption (IBAPS) scheme and give a security model for it. We also construct an IBAPS scheme and prove its security under the discrete logarithm assumption and computational Diffie–Hellman assumption. Moreover, we do an efficiency comparison with the existing identity-based signcryption schemes and anonymous signcryption schemes and show that our scheme is much more efficient than those schemes, we also compare the efficiency of our scheme with the available proxy signcryption schemes and show that our scheme provides anonymity to the proxy sender at cost less than those of existing proxy signcryption schemes.

Keywords: Identity-based cryptography, signcryption, anonymity, proxy signature, delegation of rights, provable security

MSC 2010: 94A60, 94A62, 68P25

Communicated by: Maria González Vasco

1 Introduction

Authentication and confidentiality are two fundamental requirements of public key cryptography. Digital signatures provide authentication and encryption schemes provide confidentiality. A traditional approach to provide these two security properties was to either encrypt a message and then sign the ciphertext or to sign a message and then encrypt the message and signature pair. To reduce the total computational time and communication cost of doing both compositions serially using either of the approaches: *sign-then-encrypt* or *encrypt-then-sign*, Zheng [45] introduced a new cryptographic primitive, called *signcryption*. Signcryption aims to provide the functionality of digital signature and public key encryption in a single logical step while fulfilling all the security properties. The basic idea of signcryption is to sign and encrypt data simultaneously in parallel instead of the traditional serial approach to achieve

$$\text{Cost(Signcryption)} \ll \text{Cost(Signature)} + \text{Cost(Encryption)}.$$

***Corresponding author: Vishal Saraswat:** R.C. Bose Centre for Cryptology and Cyber Security, Indian Statistical Institute, Kolkata 700108, India, e-mail: vishal.saraswat@gmail.com. http://orcid.org/0000-0001-7082-9568
Rajeev Anand Sahu: Department d'Informatique, Université Libre de Bruxelles, Brussels, Belgium, e-mail: rajeev.sahu@ulb.ac.be
Amit K. Awasthi: Group for Cryptology Research, Gautam Buddha University, Greater Noida 201308, UP, India, e-mail: awasthi.amitk@gmail.com. http://orcid.org/0000-0002-5500-0361

For example, Zheng's proposed signcryption scheme [45] saves one modular exponentiation relative to RSA sign-then-encrypt and two modular exponentiations relative to "Schnorr signature + ElGamal encryption". This reduces the cost of computation significantly, since modular exponentiation is a dominant issue in computational cost.

1.1 Applications

Signcryption soon found its way in almost every application where public key cryptography was used to provide both authentication and confidentiality. Today, it is used in a broad range of applications [8] including authenticated key recovery [29], multicast key distribution protocol [28], secure message transmission [14], encrypted e-mail authentication by firewalls [15], secure routing in mobile ad hoc networks [31], secure networking and routing [16, 20], mobile grid web services [32], secure asynchronous transfer mode (ATM) networks [46], improved secure electronic transaction (SET) [17, 50, 51], and Electronic Fund Transfer (EFT) [36].

1.2 Anonymous signcryption

With digital communication and transactions becoming an essential part of the daily lives, the collection of information by various agents about users gives rise to a plethora of privacy concerns. The privacy of users can be guarded by providing them anonymity in their online activities. The notion of anonymous encryption was formalized in [3] towards providing the anonymity of the recipient while the notion of anonymous signatures was introduced in [39] and formalized in [35] towards providing the anonymity of the sender. Boyen [4] introduced the concept of anonymity in signcryption to provide the anonymity for the sender and the receiver to an outsider, that is, an adversary other than the receiver or the sender.

But in their setting, the sender was not anonymous to the receiver. In the real world communication, many times we come up with the condition where the sender anonymity is required even from the receiver. To achieve such a goal, Huang et al. [19] introduced "*anonymous signcryption*" using ring signature [33] which allows a user to form a ring of members (including themselves) arbitrarily without collaboration of any of those ring members and then sends confidential information to a recipient so that the message can be authenticated to have been signed by a member of the ring without revealing exactly which member of the ring is the actual signer. The receiver in an anonymous signcryption scheme only knows that the message is produced by one member of a designated group but cannot know more information about actual signcrypter's identity.

1.3 Proxy signcryption

Many widely used personal communication devices such as digital assistants, hand-held computers, pagers and mobile phones come with constrained computational capacity. The lack of hardware features in these devices is a constraint towards efficiently carrying out the heavy mathematical computations required by cryptographic primitives such as digital signature. Therefore, proxy signature schemes [27] have emerged to allow off-loading of heavy computational work from a low power device to a more powerful server. Extending this primitive to signcryption, proxy signcryption [14] was introduced so that an original user/sender could authorize a proxy agent to send confidential messages to a recipient on its behalf.

A proxy signcryption scheme enables a sender, $\mathcal{O}$, also called the *designator* or *delegator*, to delegate its signing rights (without transferring the private key) to another user $\mathcal{P}$, called the *proxy sender*, to produce, on the delegator's behalf, signcryptions that can be verified by a receiver $\mathcal{R}$ under the delegator $\mathcal{O}$'s public key. For example, the director of a company may authorize the deputy director to signcrypt certain messages on his behalf and send to the employees during a certain period of his absence. The signcryptions can be

"unsigncrypted" by the recipient employees with their respective secret keys and they can be convinced of the concurrence of the director of the company.

1.4 Anonymous proxy signcryption

In the proxy signcryption setting, the proxy sender is not anonymous to the receiver and it can know the identity of the proxy as soon as it decrypts the received ciphertext. But the role of the proxy sender is that of an intermediary between the original sender and the recipient, and in an ideal proxy signcryption setting, the proxy signer should be almost invisible to the recipient and the proxy sender's identity should not to be revealed to the recipient.

Let us consider a situation where the CEO of an office wants to issue some signcrypted memos to the employees when he is away of the office. To handle the issue, he authorizes one of his subordinates the right to create signcrypted memos on his behalf such that all the employees can be convinced by the memos that the memos have been actually authorized by the CEO but no employee can figure out which subordinate has been pet to the CEO (who has created the memos on behalf of the CEO).

Though the anonymity of the original sender and the recipient in a signcryption scheme has been widely discussed, to the best of our knowledge, there is no proxy signcryption scheme which provides anonymity to the proxy from the receiver or a solution to the above and similar issues. To achieve this objective, we introduce the notion of an anonymous proxy signcryption and propose a simple and efficient algorithm to provide anonymity to the proxy sender from the recipient.

1.5 Motivation

The application of anonymous proxy signcryption can be realized in various real world scenarios, as mentioned above, where the proxy sender needs to authenticate himself/herself as a sender of an encrypted data.

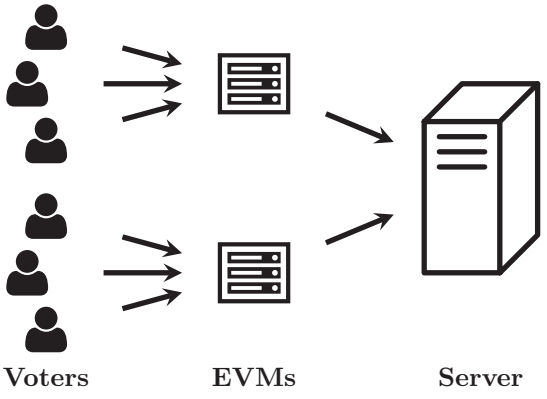

Figure 1. e-Voting.

For example, consider the case of end-to-end voting (Figure 1) through an electronic voting machine (EVM). To satisfy the requirements of *fairness*, *robustness*, *verifiability*, *confidentiality* and *privacy*, various cryptographic means are used including a combination of blind signatures and proxy signcryption. The voter acts as a user who authorizes the EVM as his/her proxy agent who proxy signcrypts for the user and then sends the signcrypted votes to the central server for counting. The EVM protocol is designed to protect the privacy of the voters so that no one is able to figure out who cast a vote for whom. But in these protocols, the identity of the EVM is not hidden and the voting distribution from a particular EVM can be found out. This information can then lead to figure out the identity of a group of users who were registered to vote at this EVM,

using geographical correlation or similar means. Such analysis has led to abuse in past [48, 49, 53] where the candidate party has threatened the voters with dire consequences mentioning this drawback of the EVM protocol which they would be able to use to find out if the majority of the voters of a particular area voted for this party or not. Knowledge of such information could be gained due to non-anonymity of the EVMs and to prevent such rampant threats and scams, the identity of the EVM itself must also be hidden, which can be achieved by using an anonymous proxy signcryption scheme in addition to the usual tools, like TOR [11], to avoid network analysis.

1.6 Related work

Since the introduction of the idea of signcryption in 1997 [45], several new schemes and improvements have been proposed. Baek et al. [1] introduced a security model for signcryption in random oracle that admits formal proofs for the confidentiality and unforgeability of signcryption. Malone-Lee [26] constructed the first identity-based signcryption scheme. Boyen [4] presented an identity-based scheme with the idea of sign-then-encrypt and provided ciphertext anonymity. The scheme in [4] is provably secure in random oracle. In 2004, Libert and Quisquater [25] modified Boyen's security model to the non-identity based signcryption setting and proposed a signcryption scheme. Unfortunately, Tan [7] showed that the scheme did not satisfy the property of ciphertext anonymity. Further using the definition proposed by Boyen [4], Chen and Lee [6] presented an improved construction of identity-based signcryption but the scheme of Barreto et al. [2], proposed in the same year, turned out to be more efficient than all the previous schemes [4, 6, 26].

To achieve sender's anonymity to the recipient in the signcryption scheme, Huang et al. [19] introduced the idea of ring signcryption based on the work of Herranz and Sáez [18]. Following the idea of ring signature for anonymity, many other identity-based ring signcryption schemes [21, 22, 30, 40–43, 47] were proposed but Selvi et al. [37] have shown that almost all the proposed ring signcryption schemes (except the one by Huang et al. [19]) and in particular [22, 40, 47] are not secure. Further, Zhang et al. [44] have shown that the scheme proposed in [42] is not anonymous from the receiver's view, and further, it is not verifiable for a third party. In 2009, Lal et al. [21] introduced the idea of anonymous identity-based signcryption for multiple receivers, the scheme was proved secure in the random oracle model. Zhang and Xu [41] formalized a security model for such schemes in the standard model. Recently, Deng et al. [10] have proposed an efficient improvement on the ID-based ring signcryption scheme.

Further, to delegate the signcryption rights to an authorized agent, Gamage et al. [14] introduced the idea of proxy signcryption by combining the concepts of proxy signature and signcryption together. But their scheme does not support a provable security. Li and Chen [23] proposed the first identity-based proxy signcryption scheme which is just a proxy variant of the Libert and Quisquater's identity-based signcryption scheme [24], the scheme proposed in [23] is not proxy protected [38]. Moreover, security of the scheme rests directly on the security of the underlying identity-based signcryption scheme [24], and no security issue of proxy delegation is considered. In 2005, Duan et al. [12] proposed the first formal model of security for the identity-based proxy signcryption schemes with delegation by warrant. In 2008, Elkamchouchi and Yasmine [13] proposed an identity-based proxy signcryption with partial delegation [27], following the construction of [24]. The scheme [13] is also not proxy protected, as due to partial delegation it is indistinguishable that the signcryption is created by either the original sender or by the proxy sender, hence their scheme cannot be considered for practical applications.

We can see that several signcryption schemes have been proposed providing anonymity to the sender, but all the available schemes rely on the ring signature [33]. It has been well studied that, due to the ring structure, computational complexity (or communication overhead) of a ring signature scheme increases unexpectedly with the large group, so it is of great interest to achieve anonymity without applying the ring signature.

1.7 Our contribution

To the best of our knowledge, there is no proxy signcryption scheme which provides anonymity to the proxy from the receiver. To achieve this objective, we introduce the notion of an anonymous proxy signcryption and propose a simple and efficient algorithm to provide anonymity to the proxy sender from the recipient.

We introduce a formal model for an identity-based anonymous proxy signcryption (IBAPS) scheme and also formalize a security model for an IBAPS scheme. We give a concrete instantiation of an IBAPS scheme and prove the security of the scheme based on the hardness of the discrete logarithm problem, the computational Diffie–Hellman problem and the bilinear Diffie–Hellman problem. In our definition, we also provide a mechanism to the original sender to expose the identity of the proxy sender in case of misuse of its authorization. The delegator needs to have enough trust on the proxy sender to authorize it to sign on its behalf but in reality, trust is broken quite regularly. We build on the technique of *pseudonym* used in a recently proposed anonymous proxy signature scheme [34] to provide the required functionality – the identity of the proxy signer is hidden but in case of misuse of the delegated rights, the original signer can reveal the proxy signer's identity.

Further we compare the efficiency of our scheme with the existing identity-based signcryption schemes and anonymous signcryption schemes and show that our scheme is much more efficient than those schemes, we also compare the efficiency of our scheme with the available proxy signcryption schemes and show that our scheme provides anonymity to the proxy sender at cost less than those of existing proxy signcryption schemes.

1.8 Outline of the paper

The rest of this paper is organized as follows. In Section 2, we introduce some related mathematical definitions, problems and assumptions. In Section 3, we present the formal definition of an IBAPS scheme and a security model for it. We present the new primitive of IBAPS scheme in Section 4. In Section 5, we analyze the security of our scheme. In Section 6, we compare the efficiency of our scheme with the other available schemes and show that our scheme is more efficient than the existing schemes. Finally, Section 7 gives a brief conclusion.

2 Preliminaries

In this section, we introduce some relevant definitions, mathematical problems and assumptions. Unless stated otherwise, all algorithms are probabilistic and polynomial-time. Further, all adversaries are polynomial-time and are allowed to make at most polynomial number of queries to the oracle(s) they have access to.

Definition 2.1 (Bilinear map). Let G_1 be an additive cyclic group with generator P and let G_2 be a multiplicative cyclic group with generator g. Let both groups be of the same prime order q. Then a map $e : G_1 \times G_1 \to G_2$ satisfying the following properties is called a *cryptographic* bilinear map:

(i) *Bilinearity*: $e(aP, bP) = e(P, P)^{ab}$ for all $a, b \in \mathbb{Z}_q^*$, or equivalently, $e(Q + R, S) = e(Q, S)e(R, S)$ and $e(Q, R + S) = e(Q, R)e(Q, S)$ for all $Q, R, S \in G_1$.

(ii) *Non-degeneracy*: There exist $Q, R \in G_1$ such that $e(Q, R) \neq 1$. Note that since G_1 and G_2 are groups of prime order, this condition is equivalent to the condition $e(P, P) \neq 1$, which again is equivalent to the condition that $e(P, P)$ is a generator of G_2.

(iii) *Computability*: There exists an efficient algorithm to compute $e(Q, R) \in G_2$ for any $Q, R \in G_1$.

Definition 2.2 (DLP). Let G_1 be a cyclic group with generator P. Given a random element $Q \in G_1$, the *discrete log problem* (DLP) in G_1 is to compute an integer $n \in \mathbb{Z}_q^*$ such that $Q = nP$.

Definition 2.3. Let G_1 be a cyclic group with generator P. The *DL assumption* on G_1 states that the probability of any polynomial-time algorithm to solve the DLP in G_1 is negligible.

Definition 2.4 (CDHP). Let G_1 be a cyclic group with generator P. Let $a, b \in \mathbb{Z}_q^*$ be randomly chosen and kept secret. Given $P, aP, bP \in G_1$, the *computational Diffie–Hellman problem* (CDHP) is to compute $abP \in G_1$.

Definition 2.5. Let G_1 be a cyclic group with generator P. The (t, ϵ)-*CDH assumption* holds in G_1 if there is no algorithm which takes at most t running time and can solve CDHP with probability at least ϵ where the probability is over the random choice of generator $P \in G_1 \setminus \{0\}$ and random choice of $a, b \in \mathbb{Z}_q^*$.

Definition 2.6 (CBDHP). Let G_1 be a cyclic group with generator P. Let $e : G_1 \times G_1 \to G_2$ be a bilinear map. Let $a, b, c \in \mathbb{Z}_q^*$ be randomly chosen and kept secret. Given $P, aP, bP, cP \in G_1$, the *computational bilinear Diffie–Hellman problem* (CBDHP) is to compute $e(P, P)^{abc} \in G_2$.

Definition 2.7. Let G_1 be a cyclic group with generator P. Let $e : G_1 \times G_1 \to G_2$ be a bilinear map. The (t, ϵ)-*CBDH assumption* holds in G_1 if there is no algorithm which takes at most t running time and can solve CBDHP with probability at least ϵ where the probability is over the random choice of generator $P \in G_1 \setminus \{0\}$ and random choice of $a, b, c \in \mathbb{Z}_q^*$.

3 Definition and security of IBAPS

In this section, we formally define an *identity-based anonymous proxy signcryption* (IBAPS) scheme and the security model for it.

3.1 Formal model for IBAPS

An IBAPS scheme comprises of six algorithms: Setup, Extract, ProxyGen, AnonProxySigncrypt, Unsigncrypt and ProxyReveal.

- $(params, s) \leftarrow \mathsf{Setup}(1^k)$: This is the system initialization algorithm run by a private key generator (PKG) which takes as input a security parameter 1^k and outputs the public (system) parameters *params*, a master secret key *msk* of the PKG and a corresponding system wide public key *Pub*. (We will include *Pub* in the *params* for brevity.)
- $(pk_{\mathrm{ID}}, sk_{\mathrm{ID}}) \leftarrow \mathsf{Extract}(\mathrm{ID}, s)$: This is the key-generation algorithm run by the PKG which takes as input a user's identity ID, the public parameters *params* and the master secret key *msk* and outputs a secret key sk_{ID} associated to the identity ID. (The public key pk_{ID} associated to the identity ID can be computed by anyone using the public parameters *params*.)
- $(w, \sigma_w, H_{\mathcal{Q}}, S_{\mathcal{Q}}) \leftarrow \mathsf{ProxyGen}(\mathrm{ID}_{\mathcal{O}}, \mathrm{ID}_{\mathcal{P}}, sk_{\mathcal{O}}, sk_{\mathcal{P}})$: This is an interactive protocol between the original sender $\mathcal{O}$ and the proxy sender $\mathcal{P}$ which takes as input their identities $\mathrm{ID}_{\mathcal{O}}$ and $\mathrm{ID}_{\mathcal{P}}$, their private keys $sk_{\mathcal{O}}$ and $sk_{\mathcal{P}}$, the public parameters *params*, and outputs
  - the *pseudonym* $H_{\mathcal{Q}}$ which anonymizes the identity of the proxy sender,
  - a signed warrant w which includes the nature of message to be delegated, period of delegation, identity information of original sender, the *pseudonym* for the proxy sender and other relevant information,
  - a delegation σ_w, and
  - the proxy signing key $S_{\mathcal{Q}}$.
- $\sigma \leftarrow \mathsf{AnonProxySigncrypt}(pk_{\mathcal{O}}, pk_{\mathcal{R}}, H_{\mathcal{Q}}, S_{\mathcal{Q}}, m)$: This is a probabilistic algorithm to produce an anonymous proxy signcryption run by the proxy sender $\mathcal{P}$ which takes as input the original sender's public key $pk_{\mathcal{O}}$, the receiver's public key $pk_{\mathcal{R}}$, the pseudonym $H_{\mathcal{Q}}$, the proxy signing key $S_{\mathcal{Q}}$, and the message m that needs to be signcrypted, the public parameters *params* and outputs a ciphertext σ.

- m or $\perp \leftarrow \mathsf{Unsigncrypt}(sk_{\mathcal{R}}, pk_{\mathcal{O}}, \sigma)$: This is a deterministic algorithm for decryption and verification run by the receiver $\mathcal{R}$ which takes as input a ciphertext σ, the recipient's secret key $sk_{\mathcal{R}}$ and the original sender's public key $pk_{\mathcal{O}}$, the public parameters *params* and outputs a message m or an invalid symbol $\perp$.
- 'true' or 'false' $\leftarrow \mathsf{ProxyReveal}(H_{\mathcal{Q}}, \mathrm{ID}_{\mathcal{P}})$: This algorithm is run by the original sender $\mathcal{O}$ to reveal the identity of the proxy sender $\mathcal{P}$. It takes as input the pseudonym $H_{\mathcal{Q}}$ and an identity $\mathrm{ID}_{\mathcal{P}}$ and outputs 'true' or 'false'.

Remark 3.1. Note that the identity $\mathrm{ID}_{\mathcal{P}}$ of the proxy sender $\mathcal{P}$ or its public key $pk_{\mathcal{P}}$ or secret key $sk_{\mathcal{P}}$ are not required for AnonProxySigncrypt or Unsigncrypt.

Definition 3.2 (Consistency). For all messages m, and for all valid key-pairs, $(pk_{\mathcal{O}}, sk_{\mathcal{O}})$, $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$ and $(H_{\mathcal{Q}}, S_{\mathcal{Q}})$, we require

$$\mathsf{Unsigncrypt}(sk_{\mathcal{R}}, pk_{\mathcal{O}}, \mathsf{AnonProxySigncrypt}(pk_{\mathcal{O}}, pk_{\mathcal{R}}, H_{\mathcal{Q}}, S_{\mathcal{Q}}, m)) = m.$$

3.2 Security model for IBAPS

An anonymous proxy signcryption scheme has four security requirements: message confidentiality, ciphertext unforgeability, proxy anonymity and accountability. We consider the strongest possible notions of confidentiality, unforgeability, anonymity and accountability for the security model of an identity based anonymous proxy signcryption scheme.

Definition 3.3 (Confidentiality). An IBAPS is said to be *indistinguishable against adaptive chosen ciphertext and adaptive chosen identity attack* (IND-IBAPS-CCA2) if no polynomial time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following IND-IBAPS-CCA2 game:

- *Initialize*: Challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$, keeps master key s secret and sends the public parameters *params* to the adversary $\mathcal{A}$.
- *Probe* 1: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Challenge*: The adversary $\mathcal{A}$ selects and gives to the challenger two identities ID_S, ID_R, a pseudonym H_Q, a warrant w and two messages m_0, m_1 (of equal length) on which it wishes to be challenged. The challenger $\mathcal{C}$ flips a fair binary coin β, signcrypts m_β under the warrant w through the pseudonym H_Q with the sender identity ID_S and receiver identity ID_R and then sends the target ciphertext σ to $\mathcal{A}$.
- *Probe* 2: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Guess*: Finally, $\mathcal{A}$ produces his guess β' on β.

$\mathcal{A}$ wins the game if $\beta' = \beta$ and $\mathcal{A}$ did not receive a response of Extract query on ID_R in *Probe* 1 or *Probe* 2 and $\mathcal{A}$ did not receive a response of Unsigncrypt query on target ciphertext σ. $\mathcal{A}$'s advantage of winning the IND-IBAPS-CCA2 game is defined to be

$$\mathrm{Adv}^{\text{IND-CCA2}}_{\text{IBAPS},\mathcal{A}}(k) = \left|[\beta' = \beta] - \tfrac{1}{2}\right|.$$

Definition 3.4 (Unforgeability). An IBAPS scheme is said to be *strongly (existentially) unforgeable against adaptive chosen-message and adaptive chosen-ID attack* (sUF-IBAPS-CMA2) if no polynomial time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger $\mathcal{C}$ in the following sUF-IBAPS-CMA2 game:

- *Initialize*: Challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$, keeps master key s secret and sends the public parameters *params* to the adversary $\mathcal{A}$.
- *Probe*: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Forge*: Finally, $\mathcal{A}$ produces two identities ID_S, ID_R, a pseudonym H_Q, and an anonymous proxy signcryption σ on some message m under a warrant w.

$\mathcal{A}$ wins the game if $m = \mathsf{Unsigncrypt}(sk_{\mathcal{R}}, pk_{\mathcal{O}}, \sigma)$ and $\mathcal{A}$ has never received during *Probe* a response σ of AnonProxySigncrypt query on message m under a warrant w from ID_S to ID_R through the proxy pseudonym

H_Q, a response of Extract query on ID_S and a response of ProxyGen query on H_Q with warrant w. $\mathcal{A}$'s advantage of winning the sUF-IBAPS-CMA2 game is defined to be

$$\mathrm{Adv}^{\text{sUF-CMA2}}_{\text{IBAPS},\mathcal{A}}(k) = \Pr[\mathcal{A} \text{ wins}].$$

Definition 3.5 (Anonymity). By *anonymity* we mean that no one except the original sender should be able to determine the identity of the proxy sender from the proxy signcryption or the warrant. An IBAPS scheme is said to be *proxy anonymous against adaptive chosen identity attacks* (IND-IBAPS-CIA2) if no polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following IND-IBAPS-CIA2 game between the adversary and a challenger $\mathcal{C}$:

- *Initialize*: Challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$, keeps master key s secret and sends the public parameters *params* to the adversary $\mathcal{A}$.
- *Probe* 1: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Challenge*: The adversary $\mathcal{A}$ selects and gives to the challenger an identity ID_S, a warrant w minus the proxy pseudonym, and two identities ID_P^0, ID_P^1 on which it wishes to be challenged. The challenger $\mathcal{C}$ flips a fair binary coin β, and generates a pseudonym proxy $H_Q \leftarrow \mathsf{ProxyGen}(\mathrm{ID}_S, \mathrm{ID}_P^\beta, sk_S, sk_P^\beta)$ and sends H_Q to $\mathcal{A}$.
- *Probe* 2: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Guess*: Finally, $\mathcal{A}$ produces his guess β' on β.

$\mathcal{A}$ wins the game if $\beta' = \beta$. We allow the adversary to have obtained all the secret keys for all identities including those of ID_P^0 and ID_P^1 except the proxy signing key corresponding to H_Q. $\mathcal{A}$'s advantage of winning the IND-IBAPS-CIA2 game is defined to be

$$\mathrm{Adv}^{\text{IND-CIA2}}_{\text{IBAPS},\mathcal{A}}(k) = \left|\Pr[\beta' = \beta] - \tfrac{1}{2}\right|.$$

Definition 3.6 (Accountability). *Accountability* ensures that the proxy sender $\mathcal{P}$ does not abuse its anonymity. The original sender should be able to prove that $\mathcal{P}$ is the sender of any valid proxy signcryption. An IBAPS scheme is said to be *accountable against adaptive chosen pseudonym attacks* (Acc-IBAPS-CQA2) if no polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following Acc-IBAPS-CQA2 game between the adversary and a challenger $\mathcal{C}$:

- *Initialize*: Challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$, keeps master key s secret and sends the public parameters *params* to the adversary $\mathcal{A}$.
- *Probe*: $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs.
- *Challenge*: Finally, $\mathcal{A}$ produces two identities ID_S, ID_R, a pseudonym H_Q, and an anonymous proxy signcryption σ on some message m under a warrant w.

$\mathcal{A}$ wins the game if $m = \mathsf{Unsigncrypt}(sk_{\mathcal{R}}, pk_{\mathcal{O}}, \sigma)$ and $\mathcal{C}$ cannot reveal the identity of the proxy signer and prove it. $\mathcal{A}$'s advantage of winning the Acc-IBAPS-CQA2 game is defined to be

$$\mathrm{Adv}^{\text{Acc-CQA2}}_{\text{IBAPS},\mathcal{A}}(k) = \Pr[\mathcal{A} \text{ wins}].$$

Remark 3.7. Note that the original sender always knows the identity of the proxy sender $\mathcal{P}$ since it delegates its rights to $\mathcal{P}$. Delegation of rights to an unconditionally anonymous proxy agent does not seem practical to us. In such cases, there must be either a trusted authority or a group manager (in a group setting) to revoke the anonymity of the proxy or there must be a ring setting. All of these have significant efficiency overheads and none of these provide unconditional anonymity anyway.

4 Our IBAPS scheme

In this section, we present our identity-based anonymous proxy signcryption (IBAPS) scheme which provides anonymity to the proxy sender while also providing a mechanism to the original sender to expose the identity of the proxy sender in case of misuse. Our scheme consists of the following phases: *setup*, *key extraction*, *proxy generation*, *anonymous proxy signcryption*, *unsigncryption* and *proxy revelation*.

4.1 Setup

For a given security parameter 1^k, the PKG defines the message space $\mathcal{M} := \{0,1\}^n$ and selects an additive cyclic group G_1 of prime order $q \approx 2^n$ with generator P, a multiplicative cyclic group G_2 of the same prime order q, and a cryptographic bilinear map $e : G_1 \times G_1 \to G_2$ as defined above. The PKG then selects five cryptographic hash functions:
(i) $H_0 : \{0,1\}^* \to G_1$,
(ii) $H_1 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$,
(iii) $H_2 : \{0,1\}^* \times G_1 \to \mathbb{Z}_q^*$,
(iv) $H_3 : G_2 \to \{0,1\}^n$,
(v) $H_4 : \{0,1\}^n \to \{0,1\}^n$.
The PKG randomly selects $s \in \mathbb{Z}_q^*$ and computes the system wide public key, $Pub = sP$. Finally, the PKG publishes system's public parameter

$$params = \langle k, n, q, G_1, G_2, e, H_0, H_1, H_2, H_3, H_4, P, Pub \rangle$$

and keeps the master secret s confidential to itself.

Remark 4.1. Note that the hash function H_2 defines a natural map on $\mathbb{Z}_q^* \times G_1$ by considering the natural embedding of $\mathbb{Z}_q^*$ into $\{0,1\}^*$ by considering the binary representation of $t \in \mathbb{Z}_q^*$. We will identify this natural map with H_2.

4.2 Key extraction

Given an identity ID, the PKG computes the hash value $H_{\mathrm{ID}} := H_0(\mathrm{ID}) \in G_1$ and returns the public and private keys for ID as follows:

- *Public key*: $pk_{\mathrm{ID}} := H_{\mathrm{ID}} \in G_1$.
- *Private key*: $sk_{\mathrm{ID}} := sH_{\mathrm{ID}} \in G_1$.

Thus, for any user, say for the original sender $\mathcal{O}$, the private key is $sk_{\mathcal{O}}$ while anyone can compute the corresponding public key $pk_{\mathcal{O}}$.

4.3 Proxy generation

To delegate the signing capability to the proxy sender $\mathcal{P}$, the original sender does the following jobs to make a signed warrant w. The warrant includes the nature of message to be delegated, period of delegation, identity information of the original sender, the public-key of the *pseudonym* for the proxy sender, etc. In successfully completion of the protocol, the proxy sender $\mathcal{P}$ gets a proxy signing key $S_{\mathcal{Q}}$.

- *Delegation generation*: (a) *Pseudonym generation*: The proxy sender $\mathcal{P}$
  - selects a nonce, η,
  - selects a random $\rho_\eta \in \mathbb{Z}_q^*$,
  - computes $U_\eta = \rho_\eta P \in G_1$,

– computes $h_\eta = H_2(\eta, U_\eta) \in \mathbb{Z}_q^*$,
– computes $V_\eta = h_\eta sk_{\mathcal{P}} + \rho_\eta Pub \in G_1$,

and sends the nonce η and its signature $\sigma_\eta = (U_\eta, V_\eta)$ to the original sender $\mathcal{O}$ through a secure anonymous channel.

The original sender $\mathcal{O}$ accepts (η, σ_η) if

$$e(P, V_\eta) = e(Pub, h_\eta pk_{\mathcal{P}} + U_\eta)$$

and rejects otherwise. The original sender $\mathcal{O}$ then computes the proxy sender's *pseudonym*

$$H_{\mathcal{Q}} = H_{\mathcal{P}} + U_\eta \quad \text{where } H_{\mathcal{P}} = H_0(\mathrm{ID}_{\mathcal{P}}),$$

and sets the corresponding public key $pk_{\mathcal{Q}} = H_{\mathcal{Q}}$ which will be included in the warrant w and will be used as the signature verification key. The original sender $\mathcal{O}$ keeps (η, σ_η) securely with him to use in case it wants to reveal the identity of the proxy sender $\mathcal{P}$.

(b) *Delegation generation*: The original sender $\mathcal{O}$

– selects a random $\rho_w \in \mathbb{Z}_q^*$,
– computes $U_w = \rho_w P \in G_1$,
– computes $h_w = H_2(w, U_w) \in \mathbb{Z}_q^*$,
– computes $V_w = h_w sk_{\mathcal{O}} + \rho_w Pub \in G_1$,

and sends the warrant w and its delegation $\sigma_w = (U_w, V_w)$ to the proxy sender $\mathcal{P}$.

- *Delegation verification*: The proxy sender $\mathcal{P}$ accepts the delegation σ_w if

$$e(P, V_w) = e(Pub, h_w pk_{\mathcal{O}} + U_w)$$

and rejects otherwise.

- *Proxy signing key generation*: After accepting delegation σ_w, $\mathcal{P}$ computes

$$sk_{\mathcal{Q}} = sk_{\mathcal{P}} + \rho_\eta Pub$$

and sets the proxy signing key $S_{\mathcal{Q}}$ as

$$S_{\mathcal{Q}} = V_w + h_w sk_{\mathcal{Q}}.$$

Remark 4.2. Note that

$$sk_{\mathcal{Q}} = sk_{\mathcal{P}} + \rho_\eta Pub = sH_{\mathcal{P}} + \rho_\eta sP = s(H_{\mathcal{P}} + \rho_\eta P) = s(H_{\mathcal{P}} + U_\eta) = sH_{\mathcal{Q}}.$$

So, $(pk_{\mathcal{Q}}, sk_{\mathcal{Q}})$ is a valid (public-key, private-key) pair.

4.4 Anonymous proxy signcryption

To signcrypt a message $m \in \{0, 1\}^n$ anonymously on behalf of the original sender $\mathcal{O}$ for a receiver $\mathcal{R}$, the proxy sender $\mathcal{P}$ selects a random $\tau \in \{0, 1\}^n$, computes $t = H_1(\tau, m) \in \mathbb{Z}_q^*$, and proceeds as below:

- $\mathcal{P}$ computes
  – $T = tP \in G_1$,
  – $x = e(Pub, tH_{\mathcal{R}}) \in G_2$, where $H_{\mathcal{R}} = H_0(\mathrm{ID}_{\mathcal{R}})$ is the public key of $\mathcal{R}$,
  – $c_1 = \tau \oplus H_3(x) \in \{0, 1\}^n$ and
  – $c_2 = m \oplus H_4(\tau) \in \{0, 1\}^n$.
- $\mathcal{P}$ computes
  – a random $r \in_{\$} \mathbb{Z}_q$,
  – $U = rP \in G_1$,
  – $h = H_2(t, U)$ (considering the binary string representation of $t \in \mathbb{Z}_q^*$) and
  – $V = hS_{\mathcal{Q}} + rPub \in G_1$.

The anonymous proxy signcryption of the message m under the warrant w by $\mathcal{P}$ on behalf of the original sender $\mathcal{O}$ for the receiver $\mathcal{R}$ is

$$\sigma = (w, c = (c_1, c_2), T, U, V, U_w).$$

4.5 Unsigncryption

On receiving an anonymous proxy signcryption $\sigma = (w, c, T, U, V, U_w)$ under a warrant w, the receiver $\mathcal{R}$ unsigncrypts as follows:

(i) $\mathcal{R}$ checks whether or not the *pseudonym* $H_{\mathcal{Q}}$ is authorized by the original sender in the warrant w. If not, return $\perp$. Continue otherwise.

(ii) $\mathcal{R}$ computes

$$\begin{aligned} x' &= e(T, sk_{\mathcal{R}}) = e(tP, sH_{\mathcal{R}}) = e(sP, tH_{\mathcal{R}}) = e(Pub, tH_{\mathcal{R}}) = x, \\ \tau' &= c_1 \oplus H_3(x') = c_1 \oplus H_3(x) = \tau, \\ m' &= c_2 \oplus H_4(\tau') = c_2 \oplus H_4(\tau) = m, \\ t' &= H_1(\tau', m') = H_1(\tau, m) = t. \end{aligned}$$

(iii) $\mathcal{R}$ checks whether or not $T = tP$. If not, return $\perp$. Continue otherwise.

(iv) $\mathcal{R}$ checks whether or not the message m conforms to the warrant w. If not, return $\perp$. Continue otherwise.

(v) $\mathcal{R}$ computes $h = H_2(t, U)$, $h_w = H_2(w, U_w)$ and verifies if the following equality holds:

$$e(P, V) = e(Pub, h(h_w(H_{\mathcal{O}} + H_{\mathcal{Q}}) + U_w) + U).$$

If yes, return the message m, otherwise return $\perp$.

4.6 Proxy revelation

To reveal the identity of the proxy sender, the original sender $\mathcal{O}$ can reveal the nonce η and its signature $\sigma_\eta = (U_\eta, V_\eta)$ and show that

$$H_{\mathcal{Q}} = H_{\mathcal{P}} + U_\eta.$$

That U_η was indeed sent by $\mathcal{P}$ is proved by verifying that (η, σ_η) is a valid (message, signature)-pair from $\mathcal{P}$.

4.7 Correctness of the proposed IBAPS scheme

Theorem 4.3. *The proposed identity-based anonymous proxy signcryption scheme is correct.*

Proof. The correctness of the "message recovery" is already demonstrated in the steps (ii) and (iii) of the unsigncryption. The correctness of the "signature" follows since

$$\begin{aligned} e(P, V) &= e(P, hS_{\mathcal{Q}} + rPub) \\ &= e(P, h(V_w + h_w sk_{\mathcal{Q}}) + rPub) \\ &= e(P, h((h_w sk_{\mathcal{O}} + \rho_w Pub) + h_w sk_{\mathcal{Q}}) + rPub) \\ &= e(Pub, h((h_w H_{\mathcal{O}} + \rho_w P) + h_w H_{\mathcal{Q}}) + rP) \\ &= e(Pub, h(h_w H_{\mathcal{O}} + U_w + h_w H_{\mathcal{Q}}) + U) \\ &= e(Pub, h(h_w(H_{\mathcal{O}} + H_{\mathcal{Q}}) + U_w) + U). \end{aligned}$$

□

5 Security analysis of the proposed IBAPS scheme

In this section, we analyze the security, anonymity and accountability of our scheme. First, we prove that the proposed IBAPS scheme is existential unforgeable against adaptive chosen-message and adaptive chosen-ID attacks, then we prove that the proposed IBAPS scheme is indistinguishable against adaptive chosen ciphertext and adaptive chosen-ID attacks, and finally we analyze the anonymity and accountability of the scheme.

We facilitate the adversary with the Hash oracle which it can query to obtain the respective hash values $(H_0, H_1, H_2, H_3, H_4)$, the Extract oracle to obtain the private keys associated to adaptively selected identities, the ProxyGen oracle on adaptively selected warrants of its choice, the AnonProxySigncrypt oracle on the adaptively selected messages and warrants of its choice, and Unsigncrypt oracle on the adaptively selected ciphertexts of its choice for polynomially bounded number of queries ($q_H, q_E, q_{\text{pg}}, q_{\text{aps}}, q_{\text{us}}$ respectively for each type of oracle). Further we allow the adversary to adaptively select the identities (identity of the original sender, the proxy sender and the receiver) and the messages on which it wishes to forge the signcryption or wishes to be challenged for indistinguishability.

Remark 5.1. We will assume that the adversary has queried all the required hash values from the Hash oracle. If the adversary does not query a required hash value from the oracle but determines the hash value on its own, then this value is as good as random and the probability that verification equality holds is less than or equal to $\frac{1}{q}$. Thus, if the verification equality holds, the probability that the adversary queried the Hash oracle for all the hash values is greater than or equal to $(1 - \frac{1}{q})$.

Remark 5.2. Without loss of generality, we will assume that the adversary has queried all the required hash values from the Hash oracle before querying the other oracles and that the adversary has queried all the required secret keys from the Extract oracle before querying the ProxyGen, AnonProxySigncrypt and Unsigncrypt oracles. So, in our counting analysis, the scalar multiplications required to output public and private keys would already be accounted for in the Hash and the Extract oracles.

Theorem 5.3. *If there exists an adversary*

$$\mathcal{A}(t, q_{H_0}, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_E, q_{\text{pg}}, q_{\text{aps}}, q_{\text{us}}, \epsilon)$$

which forges the proposed IBAPS scheme in the random oracle model, then there exists an adversary

$$\mathcal{B}(t', \epsilon')$$

which solves CDHP in time at most

$$t' \geq t + (q_{H_0} + q_E + 3q_{\text{pg}} + 5q_{\text{aps}} + 3q_{\text{us}} + 2)C_S + (q_{\text{aps}} + 3q_{\text{us}})C_P$$

with success probability at least

$$\epsilon' \geq \frac{\epsilon}{M(q_T)},$$

where C_S denotes the total counts of scalar multiplications in group G_1, C_P denotes the total number of pairing computations, and $M(q_T)$ is a polynomial in the number of queries that $\mathcal{A}$ can make to $\mathcal{B}$.

Proof. For a security parameter 1^k, let the adversary $\mathcal{B}$ be challenged to solve the CDHP for

$$\langle q, G_1, P, sP, bP \rangle,$$

where G_1 is an additive cyclic group of prime order q with generator P and $s, b \in \mathbb{Z}_q^*$. The goal of $\mathcal{B}$ is to solve CDHP by computing $sbP \in G_1$ using $\mathcal{A}$, the adversary who claims to forge our proposed IBAPS scheme. $\mathcal{B}$ simulates the security game as a challenger and interacts with $\mathcal{A}$ as follows:

Setup. $\mathcal{B}$ chooses a multiplicative cyclic group $G_2 = \langle g \rangle$ of prime order q and a bilinear map $e : G_1 \times G_1 \to G_2$ and generates the system's public parameter

$$params = \langle k, n, q, G_1, G_2, e, P, Pub = sP, H_0, H_1, H_2, H_3, H_4 \rangle$$

for security parameter 1^k, where the hash functions $H_0, H_1, \ldots, H_4$ behave as random oracles and respond to Hash queries as below.

H_0-queries. To respond to the H_0 hash function queries, $\mathcal{B}$ maintains a list $L_{H_0} = \langle \text{ID}, h, a, \gamma \rangle$. When $\mathcal{A}$ queries the H_0 hash function on some identity $\text{ID} \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

(i) If the query ID' already appears in the list L_{H_0} in some tuple $\langle \text{ID}', h', a', \gamma' \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_0(\text{ID}) = h'$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $a' \in \mathbb{Z}_q^*$ and generates a random coin $\gamma' \in \{0,1\}$ with probability $Pr[\gamma' = 0] = \lambda$, for some $\lambda \in [0,1]$ which is fixed *a priori* for all queries.
(iii) If $\gamma = 0$, $\mathcal{B}$ computes $h' = a'(bP)$ and if $\gamma' = 1$, $\mathcal{B}$ computes $h' = a'P$.
(iv) Algorithm $\mathcal{B}$ adds the tuple $\langle \text{ID}', h', a', \gamma' \rangle$ to the list L_{H_0} and responds to $\mathcal{A}$ with h'.

H_1-queries. To respond to the H_1 hash function queries, $\mathcal{B}$ maintains a list $L_{H_1} = \langle \tau, m, f_m \rangle$. When $\mathcal{A}$ requests the H_1-query for $(\tau', m') \in \{0,1\}^n \times \{0,1\}^n$ for a randomly selected n-bit element $\tau' \in \{0,1\}^n$ and a n-bit message $m' \in \{0,1\}^n$, $\mathcal{B}$ responds as follows:

(i) If the query τ', m' already appears on the list L_{H_1} in some tuple $\langle \tau', m', f_{m'} \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_1(\tau', m') = f_{m'}$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $f_{m'} \in \mathbb{Z}_q^*$ and adds the tuple $\langle \tau', m', f_{m'} \rangle$ to the list L_{H_1} and responds to $\mathcal{A}$ with $H_1(\tau', m') = f_{m'}$.

H_2-queries. To respond to the H_2 hash function queries, $\mathcal{B}$ maintains two lists $L_{H_{2(a)}} = \langle w, U_w, f_w \rangle$ and $L_{H_{2(b)}} = \langle t, U, f_t \rangle$.

When $\mathcal{A}$ requests the H_2-query on (w', U_w') for some warrant $w' \in \{0,1\}^*$ and $U_w' \in G_1$ ($\mathcal{A}$ could have computed $U_w' = \rho_w' P$, for a randomly selected integer ρ_w'), $\mathcal{B}$ responds as follows:

(i) If the query (w', U_w') already appears on the list $L_{H_{2(a)}}$ in some tuple $\langle w', U_w', f_{w'} \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_2(w', U_w') = f_{w'}$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $f_{w'} \in \mathbb{Z}_q^*$ and adds the tuple $\langle w', U_w', f_{w'} \rangle$ to the list $L_{H_{2(a)}}$ and responds to $\mathcal{A}$ with $H_2(w', U_w') = f_{w'}$.

Similarly, when $\mathcal{A}$ requests the H_2-query on (t', U') for binary string representation of some integer $t' \in Z_q^*$ and $U' \in G_1$ ($\mathcal{A}$ could have computed $U' = r'P$, for a randomly selected integer r'), $\mathcal{B}$ responds as follows:

(i) If the query (t', U') already appears on the list $L_{H_{2(b)}}$ in some tuple $\langle t', U', f_{t'} \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_2(t', U') = f_{t'}$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $f_{t'} \in \mathbb{Z}_q^*$ and adds the tuple $\langle t', U', f_{t'} \rangle$ to the list $L_{H_{2(b)}}$ and responds to $\mathcal{A}$ with $H_2(t', U') = f_{t'}$.

H_3-queries. To respond to the H_3 hash function queries, $\mathcal{B}$ maintains a list $L_{H_3} = \langle x, f_x \rangle$. When $\mathcal{A}$ requests the H_3-query for some element $x' \in G_2$, where $\mathcal{A}$ could have computed $x' = e(Pub, f_{m'} h_{\mathcal{R}})$ for $h_{\mathcal{R}}$ received from H_0-query and $f_{m'}$ received from H_1-query, $\mathcal{B}$ responds as follows:

(i) If the query x' already appears on the list L_{H_3} in some tuple $\langle x', f_{x'} \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_3(x') = f_{x'}$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $f_{x'} \in \mathbb{Z}_q^*$ and adds the tuple $\langle x', f_{x'} \rangle$ to the list L_{H_3} and responds to $\mathcal{A}$ with $H_3(x') = f_{x'}$.

H_4-queries. To respond to the H_4 hash function queries, $\mathcal{B}$ maintains a list $L_{H_4} = \langle \tau, f_\tau \rangle$. When $\mathcal{A}$ requests the H_4-query for some element $\tau' \in \{0,1\}^n$ of his choice, which he would have already chosen during the H_1-query, $\mathcal{B}$ responds as follows:

(i) If the query τ' already appears on the list L_{H_4} in some tuple $\langle \tau', f_{\tau'} \rangle$, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $H_4(\tau') = f_{\tau'}$.
(ii) Otherwise $\mathcal{B}$ picks a random integer $f_{\tau'} \in \mathbb{Z}_q^*$ and adds the tuple $\langle \tau', f_{\tau'} \rangle$ to the list L_{H_4} and responds to $\mathcal{A}$ with $H_4(\tau') = f_{\tau'}$.

Extract queries. If $\mathcal{A}$ requests a private key on identity ID', $\mathcal{B}$ responds as follows:

(i) $\mathcal{B}$ obtains the tuple $\langle \text{ID}', h', a', \gamma' \rangle$ on the list L_{H_0}.
(ii) If $\gamma' = 0$, then $\mathcal{B}$ outputs 'failure' and terminates.
(iii) If $\gamma' = 1$, then $\mathcal{B}$ responds to $\mathcal{A}$ with $S_{\text{ID}} = a' Pub \in G_1$.

Note that $\mathcal{B}$ simulates the private key S_{ID} corresponding to the identity ID correctly since $\gamma = 1$ and that the probability that $\mathcal{B}$ does not terminate is $(1 - \lambda)$.

ProxyGen queries. These queries may be either *delegation queries* or *proxy key generation queries*.

- *Delegation queries*: When $\mathcal{A}$ queries for a delegation of a warrant $w' \in \{0,1\}^*$ by an original sender $\mathcal{O}'$ to a proxy sender $\mathcal{P}'$, $\mathcal{B}$ responds as follows:
  (i) $\mathcal{B}$ maintains a list $L_{\mathrm{del}} = \langle (w, \mathcal{O}, \mathcal{P}), U_w, V_w \rangle$ and if the tuple $(w', \mathcal{O}', \mathcal{P}')$ already appears on the list L_{del} in some tuple $\langle (w', \mathcal{O}', \mathcal{P}'), U_{w'}, V_{w'} \rangle$, then $\mathcal{B}$ responds to $\mathcal{A}$ with $(U_{w'}, V_{w'})$.
  (ii) $\mathcal{B}$ obtains the tuples $\langle \mathrm{ID}_{\mathcal{O}'}, h_{\mathcal{O}'}, a_{\mathcal{O}'}, \gamma_{\mathcal{O}'} \rangle$, $\langle \mathrm{ID}_{\mathcal{P}'}, h_{\mathcal{P}'}, a_{\mathcal{P}'}, \gamma_{\mathcal{P}'} \rangle$ on the list L_{H_0}.
  (iii) If $\gamma_{\mathcal{O}'} = 0$, then $\mathcal{B}$ outputs 'failure' and terminates.
  (iv) If $\gamma_{\mathcal{O}'} = 1$, then $H_0(\mathrm{ID}_{\mathcal{O}'}) = a_{\mathcal{O}'}P$ and $sk_{\mathcal{O}'} = a_{\mathcal{O}'}Pub$, and $\mathcal{B}$ proceeds to the next step.
  (v) $\mathcal{B}$ selects a random $\rho_{w'} \in \mathbb{Z}_q^*$ and sets $U_{w'} = \rho_{w'}P \in G_1$. If $U_{w'}$ already appears in some tuple $\langle w, U_w, f_w \rangle$ in the list $L_{H_{2(a)}}$, $\mathcal{B}$ picks another $\rho_{w'} \in \mathbb{Z}_q^*$ randomly and repeats this step.
  (vi) $\mathcal{B}$ then looks up the list $L_{H_{2(a)}}$ to obtain $f_{w'} = H_2(w', U_{w'})$ and sets $V_{w'} = f_{w'}sk_{\mathcal{O}'} + \rho_{w'}Pub$.
  (vii) Finally, $\mathcal{B}$ responds to $\mathcal{A}$ with the delegation $\sigma_{w'} = (U_{w'}, V_{w'})$ and adds $\langle (w', \mathcal{O}', \mathcal{P}'), U_{w'}, V_{w'} \rangle$ to the delegation generation list L_{del}.
  Note that $\mathcal{B}$ simulates the delegation $(U_{w'}, V_{w'})$ correctly since $\gamma_{\mathcal{O}'} = 1$ and that the probability that $\mathcal{B}$ does not terminate is $(1 - \lambda)$.
- *Proxy key generation queries*: When $\mathcal{A}$ queries for a proxy signing key for signing of messages satisfying a warrant $w' \in \{0,1\}^*$ and corresponding delegation $(U_{w'}, V_{w'})$ for the original sender $\mathcal{O}'$ by a proxy sender $\mathcal{P}'$, $\mathcal{B}$ responds as follows:
  (i) $\mathcal{B}$ maintains a list $L_{\mathrm{pg}} = \langle (w, \mathcal{O}, \mathcal{P}, U_w, V_w), S_{\mathcal{P}} \rangle$ and if the tuple $(w', \mathcal{O}', \mathcal{P}', U_{w'}, V_{w'})$ already appears on the list L_{pg} in some tuple $\langle (w', \mathcal{O}', \mathcal{P}', U_{w'}, V_{w'}), S_{\mathcal{P}'} \rangle$, then $\mathcal{B}$ responds to $\mathcal{A}$ with $S_{\mathcal{P}'}$.
  (ii) $\mathcal{B}$ obtains the tuple $\langle \mathrm{ID}_{\mathcal{P}'}, h_{\mathcal{P}'}, a_{\mathcal{P}'}, \gamma_{\mathcal{P}'} \rangle$ on the list L_{H_0}.
  (iii) If $\gamma_{\mathcal{P}'} = 0$, then $\mathcal{B}$ outputs 'failure' and terminates. If $\gamma_{\mathcal{P}'} = 1$, then $H_0(\mathrm{ID}_{\mathcal{P}'}) = a_{\mathcal{P}'}P$, $sk_{\mathcal{P}'} = a_{\mathcal{P}'}Pub$, and $\mathcal{B}$ proceeds to the next step.
  (iv) $\mathcal{B}$ then looks up the list $L_{H_{2(a)}}$ to obtain $f_{w'} = H_2(w', U_{w'})$ and sets $S_{\mathcal{P}'} = V_{w'} + f_{w'}sk_{\mathcal{P}'}$.
  (v) Finally, $\mathcal{B}$ responds to $\mathcal{A}$ with the proxy signing key $S_{\mathcal{P}'}$ and adds the tuple $\langle (w', \mathcal{O}', \mathcal{P}', U_{w'}, V_{w'}), S_{\mathcal{P}'} \rangle$ to the proxy generation list L_{pg}.
  Note that $\mathcal{B}$ simulates the proxy signing key $S_{\mathcal{P}'}$ correctly since $\gamma_{\mathcal{P}'} = 1$ and that the probability that $\mathcal{B}$ does not terminate is $(1 - \lambda)$.

AnonProxySigncrypt queries. Proceeding adaptively when adversary $\mathcal{A}$ requests for a proxy signcryption on message m' satisfying a warrant w' by the proxy sender $\mathcal{P}'$ on behalf of the original sender $\mathcal{O}'$ for the receiver $\mathcal{R}'$, $\mathcal{B}$ does the following:

(i) $\mathcal{B}$ maintains a list $L_{\mathrm{aps}} = \langle (w, m, \mathcal{O}, \mathcal{P}, \mathcal{R}), c = (c_1, c_2), T, U, V, U_w \rangle$ and if the tuple $(w', m', \mathcal{O}', \mathcal{P}', \mathcal{R}')$ already appears on the list L_{aps} in some tuple $\langle (w', m', \mathcal{O}', \mathcal{P}', \mathcal{R}'), c' = (c_1', c_2'), T', U', V', U_w' \rangle$ then $\mathcal{B}$ responds to $\mathcal{A}$ with $(w', c' = (c_1', c_2'), T', U', V', U_w')$.
(ii) $\mathcal{B}$ looks up the list L_{H_0} to obtain the three tuples $\langle \mathrm{ID}_{\mathcal{O}'}, h_{\mathcal{O}'}, a_{\mathcal{O}'}, \gamma_{\mathcal{O}'} \rangle$, $\langle \mathrm{ID}_{\mathcal{P}'}, h_{\mathcal{P}'}, a_{\mathcal{P}'}, \gamma_{\mathcal{P}'} \rangle$ and $\langle \mathrm{ID}_{\mathcal{R}'}, h_{\mathcal{R}'}, a_{\mathcal{R}'}, \gamma_{\mathcal{R}'} \rangle$.
(iii) If $\gamma_{\mathcal{O}'} = 0$ or $\gamma_{\mathcal{P}'} = 0$ then $\mathcal{B}$ reports 'failure' and terminates. Otherwise, $h_{\mathcal{O}'} = a_{\mathcal{O}'}P$ and $h_{\mathcal{P}'} = a_{\mathcal{P}'}P$, and using the above algorithms to respond to Hash, Extract and ProxyGen queries, $\mathcal{B}$ proceeds as follows:
  a. $\mathcal{B}$ obtains a delegation $\sigma_{w'} = (U_{w'}, V_{w'})$ of the warrant $w' \in \{0,1\}^*$ by $\mathcal{O}'$ for $\mathcal{P}'$.
  b. $\mathcal{B}$ obtains the proxy signing key $S_{\mathcal{P}'}$ of proxy sender $\mathcal{P}'$ for signing of messages satisfying the warrant w', corresponding to the delegation $(U_{w'}, V_{w'})$.
  c. $\mathcal{B}$ selects a random $\tau' \in \{0,1\}^n$ and computes $t' = H_1(\tau', m')$, $T' = t'P$, $x' = e(Pub, t'h_{\mathcal{R}'})$, $c_1' = \tau' \oplus H_3(x')$ and $c_2' = m' \oplus H_4(\tau')$.
  d. $\mathcal{B}$ then selects a random $r' \in \{0,1\}^*$, and computes $U' = r'P$, $h' = H_2(t', U')$, and $V' = h'S_{\mathcal{P}'} + r'Pub$.
(iv) Finally, $\mathcal{B}$ responds to the adversary $\mathcal{A}$ with $\sigma' = (w', c' = (c_1', c_2'), T', U', V', U_{w'})$ and adds the tuple $\langle (w', m', \mathcal{O}', \mathcal{P}', \mathcal{R}'), c' = (c_1', c_2'), T', U', V', U_w' \rangle$ to the proxy signcryption list L_{aps}.

Note that $\mathcal{B}$ simulates the proxy signcryption $(w', c' = (c'_1, c'_2), T', U', V', U_{w'})$ correctly since $\gamma_{\mathcal{O}'} = 1$ and $\gamma_{\mathcal{P}'} = 1$ and that the probability that $\mathcal{B}$ does not terminate is $(1 - \lambda)^2$.

Unsigncrypt queries. Proceeding adaptively when adversary $\mathcal{A}$ requests for an unsigncryption for the tuple $(w', c' = (c'_1, c'_2), T', U', V', U_{w'})$ meant for a receiver $\mathcal{R}'$ sent by the proxy sender $\mathcal{P}'$ on behalf of the original sender $\mathcal{O}'$, $\mathcal{B}$ does the following:

(i) $\mathcal{B}$ maintains a list $L_{\text{us}} = \langle (w, c = (c_1, c_2), T, U, V, U_w, \mathcal{O}, \mathcal{P}, \mathcal{R}), m\rangle$ and if the tuple $(w', c' = (c'_1, c'_2), T', U', V', U'_w, \mathcal{O}', \mathcal{P}', \mathcal{R}')$ already appears on the list L_{us} in some tuple $\langle (w', c' = (c'_1, c'_2), T', U', V', U'_w, \mathcal{O}', \mathcal{P}', \mathcal{R}'), m'\rangle$, then $\mathcal{B}$ responds to $\mathcal{A}$ with m'.

(ii) $\mathcal{B}$ checks whether or not the proxy sender $\mathcal{P}'$ is authorized by the original sender in the warrant w'. If not, return $\perp$. Continue otherwise.

(iii) $\mathcal{B}$ looks up the list L_{H_0} to obtain the three tuples $\langle \text{ID}_{\mathcal{O}'}, h_{\mathcal{O}'}, a_{\mathcal{O}'}, \gamma_{\mathcal{O}'}\rangle$, $\langle \text{ID}_{\mathcal{P}'}, h_{\mathcal{P}'}, a_{\mathcal{P}'}, \gamma_{\mathcal{P}'}\rangle$ and $\langle \text{ID}_{\mathcal{R}'}, h_{\mathcal{R}'}, a_{\mathcal{R}'}, \gamma_{\mathcal{R}'}\rangle$.

(iv) If $\gamma_{\mathcal{R}'} = 0$, then $\mathcal{B}$ reports 'failure' and terminates. Otherwise, $h_{\mathcal{R}'} = a_{\mathcal{R}'}P$ and using the above algorithms to respond to Hash, Extract and ProxyGen queries, $\mathcal{B}$ proceeds as follows:

a. $\mathcal{B}$ computes $x' = e(T', sk_{\mathcal{R}'})$ and $\tau' = c'_1 \oplus H_3(x')$ and $m' = c'_2 \oplus H_4(\tau')$ and $t' = H_1(\tau', m')$.

b. $\mathcal{B}$ checks whether or not $T' = t'P$. If not, return $\perp$. Continue otherwise.

c. $\mathcal{B}$ checks whether or not the message m' conforms to the warrant w'. If not, return $\perp$. Continue otherwise.

d. $\mathcal{B}$ computes $h' = H_2(t', U')$, $h_{w'} = H_2(w', U_{w'})$ and verifies if the following equality holds:

$$e(P', V') = e(Pub, h'(h_{w'}(H_{\mathcal{O}'} + H_{\mathcal{P}'}) + U_{w'}) + U').$$

If yes, return the message m', otherwise return $\perp$.

(v) Finally, $\mathcal{B}$ adds the tuple $\langle (w', c' = (c'_1, c'_2), T', U', V', U'_w, \mathcal{O}', \mathcal{P}', \mathcal{R}'), m'\rangle$ to the unsigncryption list L_{us}.

Note that $\mathcal{B}$ simulates the unsigncryption correctly since $\gamma_{\mathcal{R}'} = 1$ and that the probability that $\mathcal{B}$ does not terminate is $(1 - \lambda)$.

Forge. If $\mathcal{B}$ never reports 'failure' in the above game, then $\mathcal{A}$ outputs a valid identity-based anonymous proxy signcryption (w, c, T, U, V, U_w) on message m which satisfies

$$e(P, V) = e(Pub, h\{h_w(H_{\mathcal{O}} + H_{\mathcal{P}}) + U_w\} + U).$$

Hence, $\mathcal{A}$ outputs a new valid identity-based anonymous proxy signcryption (w, c, T, U, V, U_w) on message m with the probability

$$(1 - \lambda)^{q_E + q_{\text{pg}} + 2q_{\text{aps}} + q_{\text{us}}}(1 - \tfrac{1}{q}).$$

Now we compute the success probability of $\mathcal{B}$ for the solution of CDHP using the above forgeries (by $\mathcal{A}$). We consider both possible cases, viz., success probability in case $\mathcal{A}$ plays against the original sender and in case $\mathcal{A}$ plays against the proxy sender.

- *Case 1*: Suppose $\mathcal{A}$ simulates $\mathcal{B}$ and requests to interact with a user, say $\text{ID}_{\mathcal{O}}$, where the user $\text{ID}_{\mathcal{O}}$ is playing the role of the original sender. For $\text{ID}_{\mathcal{O}}$, $\mathcal{A}$ did not request the private key in Extract queries, $\mathcal{A}$ did not request a ProxyGen query including $\langle w, \text{ID}_{\mathcal{O}}\rangle$ and $\mathcal{A}$ did not request an AnonProxySigncrypt query including $\langle \text{ID}_{\mathcal{O}}, w, m\rangle$. In H_0-query, if $\gamma = 0$ then $H_0(\text{ID}_{\mathcal{O}}) = a_{\mathcal{O}}(bP)$, and if $\gamma = 1$ then $H_0(\text{ID}_{\mathcal{P}}) = a_{\mathcal{P}}P$. Further $\mathcal{B}$ computes

$$V^* = V' - [f_{t'}\{f_{w'}(a_{\mathcal{P}}) + \rho_{w'}\} + r']Pub$$

then proceeds to solve CDHP using the equality

$$\begin{aligned}
e(P, V') &= e(Pub, f_{t'}\{f_{w'}(h_{\mathcal{O}} + h_{\mathcal{P}}) + U_{w'}\} + U') \\
&= e(Pub, f_{t'}\{f_{w'}(H_0(\text{ID}_{\mathcal{O}}) + H_0(\text{ID}_{\mathcal{P}})) + U_{w'}\} + U') \\
&= e(Pub, f_{t'}\{f_{w'}(H_0(\text{ID}_{\mathcal{P}})) + U_{w'}\} + U' + f_{t'}\{f_{w'}H_0(\text{ID}_{\mathcal{O}})\}) \\
&= e(Pub, [f_{t'}\{f_{w'}(a_{\mathcal{P}}P) + \rho_{w'}P\} + r'P] + [f_{t'}\{f_{w'}H_0(\text{ID}_{\mathcal{O}})\}]) \\
&= e(Pub, f_{t'}\{f_{w'}(a_{\mathcal{P}}) + \rho_{w'}\}P + r'P)e(Pub, f_{t'}\{f_{w'}H_0(\text{ID}_{\mathcal{O}})\}) \\
&= e(P, [f_{t'}\{f_{w'}(a_{\mathcal{P}}) + \rho_{w'}\} + r']Pub)e(Pub, f_{t'}\{f_{w'}H_0(\text{ID}_{\mathcal{O}})\}),
\end{aligned}$$

which, by the above, can be written as

$$\begin{aligned} e(P, V^*) &= e(Pub, f_{t'}\{f_{w'}H_0(\mathrm{ID}_{\mathcal{O}})\}) \\ &= e(Pub, f_{t'}f_{w'}a_{\mathcal{O}}(bP)) \quad \text{for } \gamma = 0 \\ &= e(P, f_{t'}f_{w'}a_{\mathcal{O}}(bsP)) \\ &= e(P, K(bsP)), \end{aligned}$$

where $K = f_{t'}f_{w'}a_{\mathcal{O}} \in \mathbb{Z}_q^*$.
Comparing the components on both sides, $\mathcal{B}$ gets

$$V^* = K(bsP)$$

which implies that $K^{-1}V^* = bsP$. Thus, $\mathcal{B}$ can solve an instance of CDHP. Further, note that the probability of success is $\lambda(1-\lambda)$.

- *Case 2*: Suppose $\mathcal{A}$ simulates $\mathcal{B}$ and requests to interact with a user $\mathrm{ID}_{\mathcal{P}}$, where user $\mathrm{ID}_{\mathcal{P}}$ is the proxy sender. For $\mathrm{ID}_{\mathcal{P}}$, $\mathcal{A}$ did not request the private key in Extract queries, $\mathcal{A}$ did not request a ProxyGen query including $\langle w, \mathrm{ID}_{\mathcal{P}}\rangle$ and $\mathcal{A}$ did not request an AnonProxySigncrypt query including $\langle \mathrm{ID}_{\mathcal{P}}, w, m\rangle$. As in the above case, we can show that $\mathcal{B}$ can derive sbP with the same success probability $\lambda(1-\lambda)$.

  Hence the overall success probability that $\mathcal{B}$ solves the CDHP in the above attack game is

$$\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}+q_{\mathrm{us}}+1}(1-\tfrac{1}{q})\epsilon.$$

Now the maximum possible value of the above probability occurs for

$$\lambda = \frac{1}{q_E + q_{\mathrm{pg}} + 2q_{\mathrm{aps}} + q_{\mathrm{us}} + 1}.$$

Hence the success probability of $\mathcal{B}$ in solving the given CDHP is

$$\epsilon' \geq \frac{\epsilon}{M(q_T)},$$

where $1/M(q_T)$ is the polynomial

$$(1-\tfrac{1}{q})\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}+q_{\mathrm{us}}+1}$$

evaluated at

$$\lambda = \frac{1}{q_E + q_{\mathrm{pg}} + 2q_{\mathrm{aps}} + q_{\mathrm{us}} + 1}.$$

Now taking care of the running time, one can observe that the running time of algorithm $\mathcal{B}$ is the same as $\mathcal{A}$'s running time plus the time taken to respond to the Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt queries, that is,

$$q_{H_0} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_E + q_{\mathrm{pg}} + q_{\mathrm{aps}} + q_{\mathrm{us}}.$$

Hence, the maximum running time is given by

$$t + (q_{H_0} + q_E + 3q_{\mathrm{pg}} + 5q_{\mathrm{aps}} + 3q_{\mathrm{us}} + 2)C_S + (q_{\mathrm{aps}} + 3q_{\mathrm{us}})C_P,$$

as each H_0 Hash query requires one scalar multiplication in G_1, the Extract query also requires one scalar multiplication in G_1, the ProxyGen query requires at most three scalar multiplications in G_1, the AnonProxySigncrypt query requires five scalar multiplications in G_1, the Unsigncrypt query requires three scalar multiplications in G_1, and to output CDH solution from $\mathcal{A}$'s forgery, $\mathcal{B}$ requires at most two scalar multiplications in G_1. Additionally, $\mathcal{B}$ needs to compute one pairing to respond the AnonProxySigncrypt query and at least three pairings to respond the Unsigncrypt query. Hence

$$t' \geq t + (q_{H_0} + q_E + 3q_{\mathrm{pg}} + 5q_{\mathrm{aps}} + 3q_{\mathrm{us}} + 2)C_S + (q_{\mathrm{aps}} + 3q_{\mathrm{us}})C_P. \qquad \square$$

Theorem 5.4. *If there exists an IND-IBAPS-CCA2 adversary*

$$\mathcal{A}(t, q_{H_0}, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_E, q_{\text{pg}}, q_{\text{aps}}, q_{\text{us}}, \epsilon)$$

which runs in time t and has an advantage ϵ in the random oracle model, then there exists an adversary

$$\mathcal{B}(t', \epsilon')$$

which solves CBDHP in time at most

$$t' \geq t + (q_{H_0} + q_E + 3q_{\text{pg}} + 5q_{\text{aps}} + 3q_{\text{us}} + 1)C_S + (q_{\text{aps}} + 3q_{\text{us}})C_P$$

with success probability at least

$$\epsilon' \geq \frac{\epsilon}{M'(q_T)},$$

where C_S denotes the total counts of scalar multiplications in group G_1, C_P denotes the total number of pairing computations and $M'(q_T)$ is a polynomial in the number of queries that $\mathcal{A}$ can make to $\mathcal{B}$.

Proof. For a security parameter 1^k, let the adversary $\mathcal{B}$ be challenged to solve the CBDHP for

$$\langle q, G_1, P, sP, bP, cP\rangle,$$

where G_1 is an additive cyclic group of prime order q with generator P and $s, b, c \in \mathbb{Z}_q^*$. The goal of $\mathcal{B}$ is to solve CBDHP by computing $e(P, P)^{sbc} \in G_2$ using $\mathcal{A}$, the adversary who claims to forge our proposed IBAPS scheme. $\mathcal{B}$ simulates the security game as a challenger and interacts with $\mathcal{A}$ as follows:

Setup. Same as in the proof of unforgeability.

Probe 1. The adversary $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs similarly to the proof of unforgeability but with one exception. For the case $\gamma_{\mathcal{R}'} = 0$ in the Unsigncrypt query, $\mathcal{B}$ responds as follows:

(i) $\mathcal{B}$ looks up the list L_{H_0} to obtain the three tuples $\langle \text{ID}_{\mathcal{O}'}, h_{\mathcal{O}'}, a_{\mathcal{O}'}, \gamma_{\mathcal{O}'}\rangle$, $\langle \text{ID}_{\mathcal{P}'}, h_{\mathcal{P}'}, a_{\mathcal{P}'}, \gamma_{\mathcal{P}'}\rangle$ and $\langle \text{ID}_{\mathcal{R}'}, h_{\mathcal{R}'}, a_{\mathcal{R}'}, \gamma_{\mathcal{R}'}\rangle$. Since $\gamma_{\mathcal{R}'} = 0$, one has $H_0(\text{ID}_{\mathcal{R}'}) = h_{\mathcal{R}'} = a_{\mathcal{R}'}(bP)$.

(ii) $\mathcal{B}$ looks up the lists L_{H_1}, L_{H_3} and L_{H_4} for tuples $(\tau', m', f_{m'})$, $(x', f_{x'})$ and $(\tau', f_{\tau'})$ which satisfy the following equations:

$$x' = e(Pub, f_{m'} h_{\mathcal{R}'}), \quad T' = f_{m'}P, \quad c_1 = \tau' \oplus f_{x'}, \quad c_2 = m' \oplus f_{\tau'}.$$

(iii) If tuples satisfying the above exist, $\mathcal{B}$ responds to $\mathcal{A}$ with m'.

(iv) If such tuples do not exist, $\mathcal{B}$ responds to $\mathcal{A}$ with $\perp$.

Note that a random tuple (c_1, c_2, T) (not in $\mathcal{B}$'s lists) forms a valid ciphertext with probability at most $\frac{1}{q}$ and the Unsigncrypt query can return $\perp$ for a ciphertext (valid from $\mathcal{A}$'s view) at most q_{us} times. Thus, the simulation of the Unsigncrypt oracle is correct with probability

$$(1 - (\tfrac{1}{2}^n))^{q_{\text{us}}} \approx (1 - \tfrac{1}{q})^{q_{\text{us}}}.$$

Challenge. The adversary $\mathcal{A}$ selects and gives to the challenger three identities $\text{ID}_{\mathcal{O}}$, $\text{ID}_{\mathcal{P}}$, $\text{ID}_{\mathcal{R}}$, a warrant w and two messages m_0, m_1 (of equal length and satisfying the warrant w) on which it wishes to be challenged, where $\mathcal{A}$ did not receive a response of the Extract query on ID_R in *Probe* 1. $\mathcal{B}$ proceeds as follows:

(i) $\mathcal{B}$ looks up the list L_{H_0} to obtain the tuple $\langle \text{ID}_{\mathcal{R}}, h_{\mathcal{R}}, a_{\mathcal{R}}, \gamma_{\mathcal{R}}\rangle$.

(ii) If $\gamma_{\mathcal{R}} = 1$, then $\mathcal{B}$ reports 'failure' and terminates. Otherwise, $h_{\mathcal{R}} = a_{\mathcal{R}}(bP)$ and $\mathcal{B}$ proceeds as follows:

a. $\mathcal{B}$ obtains $c_1^*, c_2^* \leftarrow_\$ \{0, 1\}^n$.

b. $\mathcal{B}$ obtains $U^*, V^* \leftarrow_\$ G_1$.

c. $\mathcal{B}$ looks up the delegation generation list L_{del} to obtain a valid delegation $(U_{w'}, V_{w'})$ of the warrant w' from $\mathcal{O}'$ to $\mathcal{P}'$.

d. $\mathcal{B}$ sets $T^* = (a_{\mathcal{R}}^{-1})(cP)$.

e. Finally, $\mathcal{B}$ responds to $\mathcal{A}$ with the challenge signcryption $\sigma^* = (w, (c_1^*, c_2^*), T^*, U^*, V^*, U_w^*)$.

Probe 2. The adversary $\mathcal{A}$ performs a polynomially bounded number of adaptive queries: Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt on adaptively chosen inputs similarly to *Probe* 1 with the restriction that it cannot query Extract on $\mathrm{ID}_{\mathcal{R}}$ and it cannot query Unsigncrypt on $((c_1^*, c_2^*), T^*)$.

Guess. Finally, $\mathcal{A}$ outputs its guess β'. Then $\mathcal{B}$ chooses an arbitrary x from H_3-list and outputs x as its answer to the CBDHP.

This completes the description of $\mathcal{B}$'s simulation. $\mathcal{B}$ does not report 'failure' in the above game with probability

$$\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}}.$$

Since the challenge ciphertext was chosen independent of m_β, it does not have any information of m_β. Since $\mathcal{A}$ gets a non-negligible advantage in guessing β' in the real game, in this simulation, $\mathcal{A}$'s response β' depends on three cases:

(i) If one of the H_4 queries returned a response $f_{\tau'} = c_2^* \oplus m_{\beta'}$, then this would prompt $\mathcal{A}$'s guess β' independent of any other query. The probability of such a response is $\frac{1}{2}^n$ and thus the probability that none of the H_4 queries returned a response $f_{\tau'} = c_2^* \oplus m_{\beta'}$ is

$$(1-(\tfrac{1}{2}^n))^{q_{H_4}} \approx (1-\tfrac{1}{q})^{q_{H_4}}.$$

(ii) If one of the H_1 queries returned a response $f_{m'}$ such that $T^* = f_{m'}P$ then it would be evident to $\mathcal{A}$ that the challenge encryption is not done correctly so its guess β' is random and independent of any other query. The probability of such a response is $\frac{1}{q}$ and thus the probability that none of the H_1 queries returned a response $f_{m'}$ such that $T^* = f_{m'}P$ is

$$(1-\tfrac{1}{q})^{q_{H_1}}.$$

(iii) $\mathcal{A}$ made a H_3-query for input $x^* = e(T^*, sk_{\mathcal{R}})$. Recall we are working in the case $\gamma_{\mathcal{R}} = 0$ in which case $h_{\mathcal{R}} = a_{\mathcal{R}}(bP)$ so that $sk_{\mathcal{R}} = a_{\mathcal{R}}(bPub) = a_{\mathcal{R}}(bsP)$. So,

$$x^* = e(T^*, sk_{\mathcal{R}}) = e((a_{\mathcal{R}}^{-1})(cP), a_{\mathcal{R}}(bsP)) = e(P,P)^{sbc}.$$

The probability that $\mathcal{A}$ made this query is

$$(1-(\tfrac{1}{2})^n)^{q_{H_4}}(1-\tfrac{1}{q})^{q_{H_1}}\epsilon \approx (1-\tfrac{1}{q})^{q_{H_4}+q_{H_1}}\epsilon$$

since ϵ is $\mathcal{A}$'s total advantage in guessing β.

The probability that $\mathcal{B}$'s choice x is this $x^* = e(P,P)^{sbc}$ is then $1/q_{H_3}$. Thus, the total probability that $\mathcal{B}$ returns a correct guess for $e(P,P)^{sbc}$ is

$$\begin{aligned}&(1-\tfrac{1}{q})\cdot\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}}\cdot(1-\tfrac{1}{q})^{q_{\mathrm{us}}}\cdot\frac{1}{q_{H_3}}\cdot(1-\tfrac{1}{q})^{q_{H_4}+q_{H_1}}\cdot\epsilon\\ &\quad=\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}}(1-\tfrac{1}{q})^{q_{H_1}+q_{H_4}+q_{\mathrm{us}}+1}\cdot\frac{1}{q_{H_3}}\cdot\epsilon.\end{aligned}$$

Hence the overall success probability that $\mathcal{B}$ solves an instance of CBDHP in the above attack game is

$$\epsilon' = \lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}}(1-\tfrac{1}{q})^{q_{H_1}+q_{H_4}+q_{\mathrm{us}}+1}\cdot\frac{1}{q_{H_3}}\cdot\epsilon.$$

Now the maximum possible value of the above probability occurs for

$$\lambda = \frac{1}{q_E + q_{\mathrm{pg}} + 2q_{\mathrm{aps}} + 1}.$$

Hence the success probability of $\mathcal{B}$ in solving the given CDHP is

$$\epsilon' \geq \frac{\epsilon}{M'(q_T)},$$

where $1/M'(q_T)$ is the polynomial

$$\lambda(1-\lambda)^{q_E+q_{\mathrm{pg}}+2q_{\mathrm{aps}}}(1-\tfrac{1}{q})^{q_{H_1}+q_{H_4}+q_{\mathrm{us}}+1}\cdot\frac{1}{q_{H_3}}$$

evaluated at

$$\lambda = \frac{1}{q_E + q_{\text{pg}} + 2q_{\text{aps}} + 1}.$$

Now it can be observed that the running time of algorithm $\mathcal{B}$ is the same as $\mathcal{A}$'s running time plus the time taken to respond to the Hash, Extract, ProxyGen, AnonProxySigncrypt and Unsigncrypt queries, exactly as calculated in the proof of unforgeability, i.e.

$$t + (q_{H_0} + q_E + 3q_{\text{pg}} + 5q_{\text{aps}} + 3q_{\text{us}} + 1)C_S + (q_{\text{aps}} + 3q_{\text{us}})C_P,$$

Here, in this game to output a solution of CBDHP, $\mathcal{B}$ needs to compute only one scalar multiplication to set $T^* = (a_{\mathcal{R}}^{-1})(cP)$. Hence

$$t' \geq t + (q_{H_0} + q_E + 3q_{\text{pg}} + 5q_{\text{aps}} + 3q_{\text{us}} + 1)C_S + (q_{\text{aps}} + 3q_{\text{us}})C_P. \qquad \square$$

Theorem 5.5. *The presented identity-based anonymous proxy signcryption scheme is anonymous.*

Proof sketch. Since $\rho_\eta \in \mathbb{Z}_q^*$ is random, so is $U_\eta = \rho_\eta P$. Since U_η was communicated through a secure anonymous channel, it is hidden from any adversary. So, no adversary would be able to ascertain the identity of the proxy sender from the computation $H_{\mathcal{Q}} = H_{\mathcal{P}} + U_\eta$. $\square$

Theorem 5.6. *The presented identity-based anonymous proxy signcryption scheme is accountable.*

Proof sketch. From the proof of unforgeability in Theorem 5.3, it can be observed that the proxy signcryption could have been produced only by the user holding the secret key for the pseudonym $H_{\mathcal{Q}}$. Thus, to reveal the identity of the proxy sender, the original sender $\mathcal{O}$ can reveal the nonce η and its signature $\sigma_\eta = (U_\eta, V_\eta)$ and show that

$$H_{\mathcal{Q}} = H_{\mathcal{P}} + U_\eta.$$

That U_η was indeed sent by $\mathcal{P}$ is proved by verifying that (η, σ_η) is a valid (message, signature)-pair from $\mathcal{P}$, that is,

$$e(P, V_\eta) = e(Pub, h_\eta pk_{\mathcal{P}} + U_\eta). \qquad \square$$

6 Efficiency comparison

Here, we compare the efficiency of our scheme with the other identity-based signcryption schemes [4, 13], anonymous signcryption schemes [21, 41, 42] and the proxy signcryption scheme [23]. We show that our scheme is more efficient in the sense of computation and operation time than these schemes.

For the computation of operation time, we refer to [9] where the operation time for various cryptographic operations has been obtained using MIRACL [52], a standard cryptographic library, and the hardware platform is a PIV 3 GHZ processor with 512 MB memory and the Windows XP operating system. For the pairing-based scheme, to achieve the 1024-bit RSA level security, a Tate pairing defined over the supersingular elliptic curve $E = F_p : y^2 = x^3 + x$ with embedding degree 2 was used, where q is a 160-bit Solinas prime $q = 2^{159} + 2^{17} + 1$ and p a 512-bit prime satisfying $p + 1 = 12qr$. We note that the OT for one pairing computation is 20.04 ms, for one map-to-point hash function it is 3.04 ms, for a modular exponentiation it is 5.31 ms, for one scalar multiplication it is 6.38 ms, and for a general hash function it is < 0.001 ms. To evaluate the total operation time in the efficiency comparison tables, we use the simple method from [5, 9]. In the signcryption and unsigncryption phases we compare the total number of bilinear pairings (P), map-to-point hash functions (H), modular exponentiations (E), scalar multiplications (SM), and the consequent operation time (OT) while omitting the operation time due to a general hash function which is negligible compared to the other four operations. For example, the signcryption phase of our scheme takes 1 pairing operation, 0 map-to-point hash function, 0 modular exponentiation and 5 scalar multiplications. Hence the total operation time for this phase can be calculated as: $1 \times 20.04 + 5 \times 6.38 = 51.94$ ms. Similarly, we have computed the total OT in both phases for all the schemes.

Signcryption					
Scheme	**P**	**H**	**E**	**SM**	**OT (ms)**
Boyen [4]	1	0	1	3	44.49
Zhang et al. [41]	1	0	1	$n+8$	$76.39+6.38n$
Zhang et al. [42]	1	1	0	$4n+1$	$29.46+25.52n$
Lal et al. [21]	0	0	0	$3n+1$	$6.38+19.14n$
Hassan et al. [13]	2	0	2	2	63.46
Li et al. [23]	2	0	2	2	63.46
Our scheme	1	0	0	5	51.94

Unsigncryption					
Scheme	**P**	**H**	**E**	**SM**	**OT (ms)**
Boyen [4]	4	0	0	2	92.92
Zhang et al. [41]	6	0	0	0	120.24
Zhang et al. [42]	4	1	0	n	$83.20+6.38n$
Lal et al. [21]	$3+n$	0	1	n	$65.43+26.42n$
Hassan et al. [13]	4	0	2	0	90.78
Li et al. [23]	8	0	4	0	181.56
Our scheme	3	0	0	2	72.88

Overall time					
Scheme	**P**	**H**	**E**	**SM**	**OT (ms)**
Boyen [4]	5	0	1	5	137.41
Zhang et al. [41]	7	0	1	$n+8$	$196.63+6.38n$
Zhang et al. [42]	5	2	0	$5n+1$	$112.66+31.9n$
Lal et al. [21]	$3+n$	0	1	$4n+1$	$71.81+45.56n$
Hassan et al. [13]	6	0	4	2	154.24
Li et al. [23]	10	0	6	2	245.02
Our scheme 1	4	0	0	7	124.82

Table 1. Efficiency Comparison

From the efficiency comparison in Table 1, it is clear that our scheme is computationally more efficient than the available identity-based signcryption schemes and anonymous signcryption schemes, also our scheme provides anonymity to the proxy sender at cost less than those of the available proxy signcryption schemes.

7 Conclusion

We have proposed a new cryptographic primitive *anonymous proxy signcryption* to provide anonymity to the proxy sender in a proxy signcryption scheme with a mechanism to the original sender to expose the identity of the proxy sender in case of misuse. With introduction of a formal definition of identity-based anonymous proxy signcryption (IBAPS) scheme and a security model for it, we have proposed an IBAPS scheme and have proved its security under the discrete logarithm assumption and computational Diffie–Hellman assumption. Moreover, the proposed scheme is more efficient than the existing identity-based signcryption schemes and anonymous signcryption schemes. Additionally, our scheme provides anonymity to the proxy sender at cost less than those of existing proxy signcryption schemes.

Acknowledgment: A portion of the work has been done when the first two authors were at AIMSCS, Hyderabad, India.

References

[1] J. Baek, R. Steinfeld and Y. Zheng, Formal proofs for the security of signcryption, *J. Cryptology* **20** (2007), no. 2, 203–235.

[2] P. S. L. M. Barreto, B. Libert, N. McCullagh and J.-J. Quisquater, Efficient and provably-secure identity-based signatures and signcryption from bilinear maps, in: *Advances in Cryptology – ASIACRYPT 2005*, Lecture Notes in Comput. Sci. 3788, Springer, Berlin (2005), 515–532.

[3] M. Bellare, A. Boldyreva, A. Desai and D. Pointcheval, Key-privacy in public-key encryption, in: *Advances in Cryptology – ASIACRYPT 2001*, Lecture Notes in Comput. Sci. 2248, Springer, Berlin (2001), 566–582.

[4] X. Boyen, Multipurpose identity-based signcryption, in: *Advances in Cryptology – CRYPTO 2003*, Lecture Notes in Comput. Sci. 2729, Springer, Berlin (2003), 383–399.
[5] X. Cao, W. Kou and X. Du, A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges, *Inform. Sci.* **180** (2010), no. 15, 2895–2903.
[6] L. Chen and J. Malone-Lee, Improved identity-based signcryption, in: *Public Key Cryptography – PKC 2005*, Lecture Notes in Comput. Sci. 2286, Springer, Berlin (2005), 362–379.
[7] T. Chik-How, On the security of signcryption scheme with key privacy, *IEICE Trans. Fundam. Electron. Comm. Comput. Sci.* **88** (2005), no. 4, 1093–1095.
[8] Y. Cui and G. Hanaoka, Applications of signcryption, in: *Practical Signcryption*, Springer, Berlin (2010), 241–256.
[9] H. Debiao, C. Jianhua and H. Jin, An id-based proxy signature schemes without bilinear pairings, *Ann. Telecommun.* **66** (2011), no. 11–12, 657–662.
[10] L. Deng, C. Liu and X. Wang, An improved identity-based ring signcryption scheme, *Inf. Secur. J.* **22** (2013), no. 1, 46–54.
[11] R. Dingledine, N. Mathewson and P. Syverson, Tor: The second-generation onion router, in: *Proceedings of the 13th Conference on USENIX Security Symposium – Volume 13*, USENIX Association, Berkeley (2004), 303–320.
[12] S. Duan, Z. Cao and Y. Zhou, Secure delegation-by-warrant id-based proxy signcryption scheme, in: *Proceedings of the 2005 International Conference on Computational Intelligence and Security*, Lecture Notes in Comput. Sci. 3802, Springer, Berlin (2005), 445–450.
[13] H. Elkamchouchi and Y. Abouelseoud, A new proxy identity-based signcryption scheme for partial delegation of signing rights, IACR Cryptology ePrint Archive (2008), https://eprint.iacr.org/2008/041.pdf.
[14] C. Gamage, J. Leiwo and Y. Zheng, An efficient scheme for secure message transmission using proxy-signcryption, preprint (1999).
[15] C. Gamage, J. Leiwo and Y. Zheng, Encrypted message authentication by firewalls, in: *Public Key Cryptography – PKC'99*, Lecture Notes in Comput. Sci. 1560, Springer, Berlin (1999), 69–81.
[16] C. Gamage and Y. Zheng, Secure high speed networking with ABT and signcryption, unpublished Manuscript (1997).
[17] G. Hanaoka and Y. Zheng, Improving the secure electronic transaction protocol by using signcryption, *IEICE Trans. Fundam. Electron. Comm. Comput. Sci.* **84** (2001), no. 8, 2042–2051.
[18] J. Herranz and G. Sáez, A provably secure id-based ring signature scheme, IACR Cryptology ePrint Archive (2003), https://eprint.iacr.org/2003/261.ps.
[19] X. Huang, W. Susilo, Y. Mu and F. Zhang, Identity-based ring signcryption schemes: Cryptographic primitives for preserving privacy and authenticity in the ubiquitous world, in: *19th International Conference on Advanced Information Networking and Applications – AINA 2005*, IEEE Press, Piscataway (2005), 649–654.
[20] E. Kim, K. Nahrstedt, L. Xiao and K. Park, Identity-based registry for secure interdomain routing, in: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security – ASIACCS 2006*, ACM Press, New York (2006), 321–331.
[21] S. Lal and P. Kushwah, Anonymous id based signcryption scheme for multiple receivers, IACR Cryptology ePrint Archive (2009), https://eprint.iacr.org/2009/345.pdf.
[22] F. Li, S. Masaaki and T. Tsuyoshi, Analysis and improvement of authenticatable ring signcryption scheme, *J. Shanghai Jiaotong Univ. Sci.* **13** (2008), no. 6, 679–683.
[23] X. Li and K. Chen, Identity based proxy-signcryption scheme from pairings, in: *IEEE International Conference on Services Computing – SCC 2004*, IEEE Press, Piscataway (2004), 494–497.
[24] B. Libert and J.-J. Quisquater, A new identity based signcryption scheme from pairings, in: *IEEE Information Theory Workshop*, IEEE Press, Piscataway (2003), 155–158.
[25] B. Libert and J.-J. Quisquater, Efficient signcryption with key privacy from gap Diffie–Hellman groups, in: *Public Key Cryptography – PKC 2004*, Lecture Notes in Comput. Sci. 2947, Springer, Berlin (2004), 187–200.
[26] J. Malone-Lee, Identity-based signcryption, IACR Cryptology ePrint Archive (2002), https://eprint.iacr.org/2002/098.pdf.
[27] M. Mambo, K. Usuda and E. Okamoto, Proxy signatures: Delegation of the power to sign messages, *IEICE Trans. Fundam. Electron. Comm. Comput. Sci.* **79** (1996), no. 9, 1338–1354.
[28] K. Matsuura, Y. Zheng and H. Imai, Compact and flexible resolution of CBT multicast key-distribution, in: *Worldwide Computing and Its Applications – WWCA'98*, Lecture Notes in Comput. Sci. 1368, Springer, Berlin (1998), 190–205.
[29] T. Nishioka, K. Matsuura, Y. Zheng and H. Imai, A proposal for authenticated key recovery system, in: *Joint Workshop on Information Security and Cryptology – JW-ISC'97*, KIISC, Seoul (1997), 19–30.
[30] L. Pang, H. Li, L. Gao and Y. Wang, Completely anonymous multi-recipient signcryption scheme with public verification, *PLOS ONE* **8** (2013), DOI 10.1371/journal.pone.0063562.
[31] B.-N. Park and W. Lee, Ismanet: A secure routing protocol using identity-based signcryption scheme for mobile ad-hoc networks, *IEICE Trans. Commun.* **88** (2005), no. 6, 2548–2556.
[32] N. Park, K. Moon, K. Chung, D. Won and Y. Zheng, A security acceleration using XML signcryption scheme in mobile grid web services, in: *Proceedings of the 5th International Conference on Web Engineering – ICWE'05*, Springer, Berlin (2005), 191–196.
[33] R. L. Rivest, A. Shamir and Y. Tauman, How to leak a secret, in: *Advances in Cryptology – ASIACRYPT 2001*, Lecture Notes in Comput. Sci. 2248, Springer, Berlin (2001), 552–565.

[34] V. Saraswat and R. A. Sahu, A secure anonymous proxy multi-signature scheme, in: *11th International Conference on Security and Cryptography – SECRYPT 2014*, SciTePress, Setúbal (2014), 55–66.
[35] V. Saraswat and A. Yun, Anonymous signatures revisited, in: *Provable Security – ProvSec 2009*, Lecture Notes in Comput. Sci. 5848, Springer, Berlin (2009), 140–153.
[36] M. Seo and K. Kim, Electronic funds transfer protocol using domain-verifiable signcryption scheme, in: *Information Security and Cryptology – ICISC '99*, Lecture Notes in Comput. Sci. 1787, Springer, Berlin (2000), 269–277.
[37] S. Sharmila Deva Selvi, S. Sree Vivek and C. Pandu Rangan, On the security of identity based ring signcryption schemes, in: *Information Security – ISC 2009*, Lecture Notes in Comput. Sci. 5735, Springer, Berlin (2009), 310–325.
[38] M. Wang, H. Li and Z. Liu, Efficient identity based proxy-signcryption schemes with forward security and public verifiability, in: *Networking and Mobile Computing*, Lecture Notes in Comput. Sci. 3619, Springer, Berlin (2005), 982–991.
[39] G. Yang, D. S. Wong, X. Deng and H. Wang, Anonymous signature schemes, in: *Public Key Cryptography – PKC 2006*, Lecture Notes in Comput. Sci. 3958, Springer, Berlin (2006), 347–363.
[40] Y. Yu, F. Li, C. Xu and Y. Sun, An efficient identity-based anonymous signcryption scheme, *Wuhan Univ. J. Nat. Sci.* **13** (2008), no. 6, 670–674.
[41] B. Zhang and Q. Xu, An id-based anonymous signcryption scheme for multiple receivers secure in the standard model, in: *Advances in Computer Science and Information Technology*, Lecture Notes in Comput. Sci. 6059, Springer, Berlin (2010), 15–27.
[42] J. Zhang, S. Gao, H. Chen and Q. Geng, A novel id-based anonymous signcryption scheme, in: *Advances in Data and Web Management*, Lecture Notes in Comput. Sci. 5446, Springer, Berlin (2009), 604–610.
[43] M. Zhang, B. Yang, S. Zhu and W. Zhang, Efficient secret authenticatable anonymous signcryption scheme with identity privacy, in: *Proceedings of the IEEE ISI 2008 PAISI, PACCF, and SOCO International Workshops on Intelligence and Security Informatics*, Springer, Berlin (2008), 126–137.
[44] M. Zhang, Y. Zhong, P. Li and B. Yang, Analysis and enhance of anonymous signcryption scheme, IACR Cryptology ePrint Archive (2009), https://eprint.iacr.org/2009/194.pdf.
[45] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) ≤ cost(signature) + cost(encryption), in: *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Comput. Sci. 1294, Springer, Berlin (1997), 165–179.
[46] Y. Zheng and H. Imai, Compact and unforgeable key establishment over an ATM network, in: *Proceedings of the 17th Annual Joint Conference of the IEEE Computer and Communications Societies – INFOCOM '98*, IEEE Press, Piscataway (1998), 411–418.
[47] Z. Zhu, Y. Zhang and F. Wang, An efficient and provable secure identity-based ring signcryption scheme, *Comput. Stand. Interfaces* **31** (2009), no. 6, 1092–1097.
[48] Firstpost, Vote for Sule or no water: Did Ajit Pawar threaten Baramati voters?, preprint 2014, http://www.firstpost.com/politics/vote-for-sule-or-no-water-did-ajit-pawar-threaten-baramati-voters-1485739.html.
[49] Mail Online, Baramati residents claim politicians have threatened to cut off their water if they don't vote for Pawar candidate, preprint 2014, http://www.dailymail.co.uk/indiahome/indianews/article-2608989/Baramati-residents-claim-politicians-threatened-cut-water-dont-vote-Pawar-candidate.html.
[50] MasterCard and Visa, Secure electronic transaction specification – Book 1: Business description, 1997.
[51] MasterCard and Visa, Secure electronic transaction specification – Book 2: Programmer's guide, 1997.
[52] MIRACL, Multiprecision integer and rational arithmetic cryptographic library, http://certivox.org/display/EXT/MIRACL.
[53] The Telegraph, Vote for my cousin or we'll cut off your water, minister warns Indian farmers, preprint 2014, http://www.telegraph.co.uk/news/worldnews/asia/india/10775156/Vote-for-my-cousin-or-well-cut-off-your-water-minister-warns-Indian-farmers.html.