Journal of Universal Computer Science, vol. 24, no. 7 (2018), 892-915
submitted: 20/4/17, accepted: 12/4/18, appeared: 28/7/18

Proving Anonymity for BILMIX

Andrea Huszti
(University of Debrecen, Hungary
huszti.andrea@inf.unideb.hu)

Zita Kovács
(University of Debrecen, Hungary
kovacs.zita@inf.unideb.hu)

Abstract: A reductionist proof for sender anonymity of an asymmetric bilinear pairing based mixnet (BILMIX) is presented. We give an experiment-based definition for anonymity and show that BILMIX possesses anonymity in the semi-honest model against static adversaries assuming that the co-Bilinear Diffie-Hellman Problem, the Matching Find-Guess Problem and the Matching Diffie-Hellman Problem are hard. A new problem called Divisible Decisional Factorized Diffie-Hellman Problem (DDF-DHP) is introduced and showed that finding connection between data stored by the Registration Authority and the receiver is at least as hard as breaking DDF-DHP, with the assumption that secret keys of the Registration Authority and the special bulletin board are kept secret.
Key Words: mixnet, anonymity, asymmetric bilinear pairings, co-Bilinear Diffie-Hellman Problem, Matching Find-Guess Problem, Matching Diffie-Hellman Problem, Divisible Decisional Factorized Diffie-Hellman Problem
Category: E.3

1 Introduction

1.1 Motivation

Mix networks (or in short mixnets) ([Danezis et al. 2009], [Ren and Wu 2010], [Sampigethaya and Poovendran 2006]) are among the most widely used constructions for providing anonymous communication between participants. In 1981, shortly after the public key cryptography was presented, Chaum created a mail system which included a single computer, called *mix*. The purpose of a mix was to hide the correspondences between its input and output ([Chaum 1981]). In the same paper, a series of mixes (or cascade mix) was also proposed by increasing the number of mixes. The main idea is that each mix accepts an input batch of encrypted messages and produces an output batch containing the cryptographically transformed, permuted input batch. The cryptographic transformation is usually a re-encryption or a decryption. In this way the mixnet achieves untraceability between the input and output batches.

In real world there are many applications where providing unlinkability of the message and its sender is necessary. For example, we can think of electronic voting, electronic exam, electronic tender, electronic auction and electronic opinion poll systems, or collectively *e-assessment systems*. Although the users' identities could be known in a mass (namely the anonymity set), during the process the users should not be able to be linked with their messages. Focusing on these e-assessment systems one can observe that only the users are anonymous, the authority receiving the messages is publicly known.

After a certain deadline, the authorities evaluate the submissions and announce the results. Considering an e-exam, an e-tender or an e-auction system, at some point the users should be informed about their success, hence knowing their real identity is essential. The other reason to reveal user identity is to avoid anonymous criminal activities. The situation when a user is not cooperating in this phase, *i.e.* does not want to reveal his/her identity (e.g. undesirable grades) should also be considered. Therefore the possibility of *anonymity revocation* is crucial.

Depending on the application a mixnet should be able to efficiently handle short and long messages. In case of e-voting or e-auction schemes the votes or the bids are usually very short messages, but in case of e-exam or e-tender schemes the submission could be significantly longer. Our aim is to construct a mixnet which is able to handle *messages with arbitrary length* efficiently.

Furthermore, it could be also a natural demand that the anonymous senders receive a *reply* to their messages, at least a receipt about the successful submission. For example, in case of e-tender schemes the authority can warn the anonymous senders about a missing document. Similarly, in case of e-exams or e-poll systems having a possibility of a reply enables to proceed more than one round (e.g. the second questionnaire is chosen depending on the answers of the first one).

Another requirement for the senders is to be *eligible* which means that the user has to fulfill all the prerequisites for the participation in an e-assessment system. Only eligible users' messages are considered and evaluated and the system has to detect the users that are not eligible and reject their submissions. Since the users are anonymous, to determine their eligibility is cumbersome. The identification and the submission process should be separated, therefore the authorities of these processes as well.

The proposed mixnet is a general solution, which accomplishes all the requirements mentioned above: provides the anonymity for the users, the eligibility verification, the possibility of anonymous reply and anonymity revocation, and efficiently handles arbitrarily long messages. Moreover, considering an e-assessment system the submissions should be secret and undeniable.

1.2 Related work

Mix networks are the basis for many applications, especially in the field of electronic voting ([Jakobsson et al. 2002]), anonymous email ([Danezis et al. 2003]) and location privacy ([Golle et al. 2004, Huang et al. 2006]).

The onion routers (or in short ORs) were developed based on mixnets to provide low-latency in private communication applications such as web search or instant messaging. However, the onion routing systems have limitations in case of anonymity and they are vulnerable to some attacks: traffic analysis attacks ([Raymond 2001], [Erdin et al. 2015]) as well as intersection attacks ([Danezis and Serjantov 2004], [Wright et al. 2003]). A significant number of ORs were proposed and applied ([Backes et al. 2012], [Kate et al 2007], [Chen et al. 2015]). The largest and widely used OR system is Tor, which has more than 2 million users and thousands of onion routers ([Tor project 2003]). In 2017, a new variant of fixed-cascade mixing networks, the cMix protocol was proposed by Chaum and others ([Chaum et al. 2017]). It has a precomputation phase to avoid computationally intensive public-key cryptographic operations in its core real-time protocol. Therefore, it is low-latency and its fixed cascade structure of mixnodes yields strong anonymity. The phases of cMix are similar to the ones of onion routing, however cMix resists the typical attacks of ORs. cMix is the first mixing suitable for low latency chat for lightweight devices.

Hybrid mix networks, introduced by Pfitzmann ([Pfitzmann and Waidner 1985]), efficiently handle messages with arbitrary length by combining symmetric and asymmetric cryptographic primitives. A recent system called Riffle ([Kwon et al. 2016]) is a bandwidth and computation efficient communication system with strong anonymity and provides both sender and receiver anonymity by using verifiable shuffles and private information retrieval. The hybrid shuffle applying asymmetric encryptions is performed only once to share symmetric authenticated encryption keys. The hybrid mix designed by Ohkubo and Abe ([Ohkubo and Abe 2000]) uses symmetric encryption keys derived by applying a hash function to the results of a Diffie-Hellman key exchange. Our construction also applies a Diffie-Hellman key exchange and an asymmetric bilinear pairing for a secure symmetric key exchange.

Determining the real identity of an anonymous user by a Trusted Entity is the most commoly used technique with the directive that it uses this ability only if it is necessary and/or it has the right to do so ([Camenisch and Lysyanskaya 2001], [Chen et al. 2011], [Preneel et al. 2003], [Federrath et al. 2006]). It is often combined with blind or fair blind signatures. In [Preneel et al. 2003] Preneel and his co-workers gave Crowd-like and OR-like solutions of anonymity revocation in case of anonymous internet access. They introduced a management entity and a trustee to their proposals as well, where the trustee participates only if the revocation is needed, it does not take part in the anonymization process.

There are mixnet solutions that provide anonymous reply. In [Chaum 1981] Chaum proposed untraceable return addresses which allow the receiver to send a reply message without knowing senders identity. Another example is the Mixminion ([Danezis et al. 2003]), which is an anonymous remailer protocol and it supports Single-Use Reply Blocks (or SURBs) to allow anonymous recipients. In these schemes the sender recursively encrypts the return address block and sends it in the body of the message. These encryptions are necessary, even if the receiver does not intend to reply. In our construction cryptographic operations are needed only if the receiver sends messages back.

Our proposed protocol is based on bilinear pairings ([Menezes 1993]). As a consequence of the Boneh and Franklin's ID-based cryptosystem based on bilinear pairings ([Boneh and Franklin 2001]) many new cryptographic constructions appeared. In 2009, Zhong proposed an identity-based, re-encryption mix network ([Zhong 2009]) based on symmetric bilinear maps. Zhong's scheme applies only asymmetric encryptions, hence it can be used for sending short messages, only. At a fixed security level, group elements in the asymmetric setting are smaller and pairings can be computed more efficiently. As far as we know, our construction is the first hybrid mixnet, which is based on asymmetric bilinear maps.

1.3 Our results

The most important requirement for the mix networks is the anonymity property, *i.e.* possessing all the messages sent to determine the identity of a sender should be a hard problem. However, the possibility of anonymity revocation, anonymous reply, eligibility verification is also expected in practice, see the examples above. Most of the cases the system should also efficiently handle arbitrarily long messages, therefore, our main objective is to construct a complex mixnet which possesses all the previous requirements.

We presented our *symmetric* bilinear pairing based hybrid mixnet with anonymity revocation in 2015 ([Huszti and Kovacs 2015]). We designed a *hybrid mix* to handle short and long messages. In [Huszti and Kovacs 2015] besides describing the protocol we examined the time and space complexity compared to Zhong's proposal ([Zhong 2009]) as well. Besides the complexity calculations we also proved that our solution was correct. More details can be found in our previous paper [Huszti and Kovacs 2015].

Here, we improve our scheme ([Huszti and Kovacs 2015]) by applying *asymmetric* bilinear maps and prove sender anonymity in the semi honest model against a static adversary. In this model the corrupted parties do not deviate from the protocol specification, but they collaborate with the adversary to gather information and secrets. We assume that at least one mix server and two users are trustworthy, so they are not corrupted by the adversary, *i.e.* they do not reveal their secrets (the secret keys and secret permutations). We also assume the existence of a special bulletin board operating honestly, possessing a key pair. Furthermore, the secret keys of the Registration Authority and the bulletin board are kept secret, *e.g.* they are secretly shared in a threshold manner. We give the definition of *anonymity* with the help of an experiment and apply a reductionist proof. We call our asymmetric bilinear pairing based mixnet protocol BILMIX.

We show that if an adversary is able to break sender anonymity, then there exists a polynomial time algorithm that solves the co-Bilinear Diffie-Hellman Problem (co-BDHP), or else it solves either the Matching Find-Guess Problem (MFGP) or the Matching Diffie-Hellman Problem (MDHP). We also define variations of a new problem called *Divisible Decisional Factorized Diffie-Hellman Problem* (DDF-DHP) and show that finding connection between data stored by the Registration Authority and the receiver is at least as hard as breaking DDF-DHP.

1.4 Outline of the paper

The remainder of the paper is organized as follows. Section 2 outlines the necessary definitions and problems. Section 3 describes our proposed protocol. The security analysis with a focus on proving anonymity property is presented in section 4. Finally, Section 5 concludes with a summary.

2 Preliminaries

In this section we overview the basic definitions and the hard problems we apply for the construction of BILMIX. Our protocol is based on asymmetric bilinear maps, we apply a blind signature scheme for hiding the link between senders and their messages. The security of the protocol is based on the variations of the Diffie-Hellman problem.

Beginning with the work of Joux ([Joux 2004]) in 2000, bilinear pairings have been extensively used to design cryptographic protocols ([Boneh and Franklin 2001], [Boneh et al. 2002], [Boldyreva 2003]). We differentiate symmetric and asymmetric bilinear maps. First we give the definition of the asymmetric bilinear map.

Definition 1 Asymmetric bilinear map. Let G_1, G_2 and G_T be three groups of order q for some large prime q. A map $e : G_1 \times G_2 \to G_T$ is an asymmetric bilinear map if satisfies the following properties:

1. *Bilinear*: We say that a map $e : G_1 \times G_2 \to G_T$ is bilinear if $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$ for all $(P_1, P_2) \in G_1 \times G_2$ and all $a, b \in Z_q^*$.
2. *Non-degenerate*: The map does not send all pairs in $G_1 \times G_2$ to the identity in G_T. $\forall P_1 \in G_1$, $e(P_1, P_2) = 1$ $\forall P_2 \in G_2$ iff $P_1 = 1_{G_1}$.

3. *Computable*: There is an efficient algorithm to compute $e(P_1, P_2)$ for any $(P_1, P_2) \in G_1 \times G_2$.

Asymmetric pairings for which an efficiently-computable isomorphism $\psi : G_2 \rightarrow G_1$ is known are called *Type 2* pairings ([Galbraith et al. 2008]), while asymmetric pairings for which no efficiently computable isomorphism is known either from G_1 to G_2 or from G_2 to G_1 are called *Type 3* pairings ([Galbraith et al. 2008]). Many cryptographic protocols in the asymmetric setting rely on the existence of ψ for their security reduction while some use it in the protocol itself. Known examples of such pairings are the Weil and Tate pairings over suitable elliptic curve groups G_1 and G_2. If $G_1 = G_2$ then the pairing is symmetric and it is called a *Type 1* pairing. Since *Type 1* pairings are quite restricted in terms of the choice of curves and are significantly slower than their asymmetric counterparts at higher security levels ([Hankerson et al. 2008]), we apply pairings *Type 3*. Typically, G_1, G_2 are elliptic curve groups and G_T is a multiplicative group of a finite field.

From now on let G_1, G_2 be elliptic curve groups with generator elements P_1 and P_2, respectively. Usually the security of cryptographic protocols applying bilinear maps is based on the variants of the Diffie-Hellman Problem. In [Smart and Vercauteren 2005], the following computational and decisional problems are defined.

We define a pairing problem instance to be a tuple $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$. First we define various notions of the Computational Diffie-Hellman (CDH) and the Decisional Diffie-Hellman (DDH) problems.

Definition 2 The $CDH_{i,j,k}$ Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ and values $i, j, k \in \{1, 2\}$ we define the $CDH_{i,j,k}$ Problem to be the following: Given aP_i and bP_j, with $a, b \in Z_q^*$, we are asked to compute abP_k.

Definition 3 The $DDH_{i,j,k}$ Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ and values $i, j, k \in \{1, 2\}$ we define the $DDH_{i,j,k}$ Problem to be the following: Given aP_i, bP_j and cP_k, with $a, b, c \in Z_q^*$, we are asked to decide whether $cP_k = abP_k$.

When $G_1 = G_2$, these problems reduce to the standard CDH and DDH problems. In some publications ([Boneh et al. 2002],[Chatterjee et al. 2010]) the cases of $i = k = 1, j = 2$ are defined as co-CDH and co-DDH problems.

Similarly to what happens in symmetric pairing groups, the $DDH_{i,j,k}$ with $i \neq j$ problem is easy in asymmetric bilinear map groups. According to the terminology, those groups are called Gap Diffie-Hellman groups, where CDHP is hard, but DDHP is easy.

We can also formalize a number of variations of the Bilinear Diffie-Hellman (BDH) problem.

Definition 4 The $BDH_{i,j,k}$ Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ and values $i, j, k \in \{1, 2\}$ we define the $BDH_{i,j,k}$ problem to be the following: Given aP_i, bP_j and cP_k, with $a, b, c \in Z_q^*$, we are asked to compute $e(P_1, P_2)^{abc}$.

Variations of the co-Bilinear Diffie-Hellman (co-BDH) Problem are formalized as follows.

Definition 5 The $co - BDH_{j,k}$ Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ and values $j, k \in \{1, 2\}$ we define the $co - BDH_{j,k}$ Problem to be the following: Given aP_1, aP_2, bP_j and cP_k, with $a, b, c \in Z_q^*$, we are asked to compute $e(P_1, P_2)^{abc}$.

We apply blind short signatures to provide a valid signature on the submission in a way that the signer (Registration Authority) does not learn any information about the message. Boldyreva in [Boldyreva 2003] provided a blind signature scheme based on any Gap Diffie-Hellman (GDH) group.

We apply a blind GDH signature scheme based on the variant of the BLS signature ([Boneh et al. 2002]), the BLS-3b signature given in [Chatterjee et al. 2010], which uses *Type 3* pairings.

Definition 6 Blind BLS-3b signature scheme. The public parameters are: Gap co-Diffie-Hellman groups (G_1, G_2) with prime order q, generator elements $P_1 \in G_1$ and $P_2 \in G_2$ and a Map-to-point hash function $H : \{0, 1\}^* \to G_1$. The Blind BLS-3b signature description is the following:

- *Keygen*: The secret key is a random value $x \in Z_q^*$ and the public key is $(P_{pub_1}, P_{pub_2}) = (xP_1, xP_2) \in G_1 \times G_2$ for a signer.
- *Blind Signature Issuing Protocol*: Given secret key x and a message $m \in \{0, 1\}^*$.
  - (Blinding) The user chooses randomly $r \in Z_q^*$, computes $M' = rP_1 + H(m)$ and sends M' to the signer.
  - (Signing) The signer computes $\sigma' = xM'$ and sends back σ' to the user.
  - (Unblinding) The user then computes the signature $\sigma = \sigma' - rP_{pub_1}$ and outputs (m, σ).
- *Verify*: Given public key (P_{pub_1}, P_{pub_2}), a message m and a signature σ, verify $e(H(m), P_{pub_2}) = e(\sigma, P_2)$.

The security of the BLS-3b signature scheme is based on the variant of the Computational co-Diffie-Hellman Problem that is given in [Chatterjee et al. 2010].

Definition 7 The co-CDH* Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ we define the $co - CDH^*$ Problem to be the following: Given aP_1, aP_2 and bP_1, with $a, b \in Z_q^*$, we are asked to compute abP_1.

The intractability of discrete logarithm problem in G_1 and G_2 are both necessary for the hardness of co-CDHP*. If co-CDHP* in (G_1, G_2) is hard, and H is a random function, then the BLS-3b signature scheme is secure.

3 The BILMIX

In this section we detail the steps of BILMIX. Our proposed protocol can be built on any G_1, G_2, G_T groups, where (G_1, G_2) are Gap Diffie-Hellman groups and G_T is a multiplicative group. There are several senders $(S_1, \ldots, S_n)$, mix servers $(\mathcal{M}_1, \ldots, \mathcal{M}_N = R)$, where the last mix server is the receiver, furthermore there is a Registration Authority ($\mathcal{RA}$) participating in our protocol. We use a publicly readable special bulletin board ($\beta\beta$) for showing the verification values.

There are seven phases in BILMIX. Phases 1-5 are required, and phases 6-7 are optional. We describe them briefly at first.

Preparation: In this phase all the parameters and keys are generated, public ones are made public. A key pair is generated for $\beta\beta$. The Registration Authority generates the system parameters and a key pair, the mix servers also compute their secret, public keys. The mixnet, *i.e.* all mix servers together generate the mixnet public keys, too.

Registration: The sender indicates his intention of sending a message to the receiver and gets the permission from the Registration Authority for it. If the sender is eligible then the Registration Authority gives a signature on the sender's blinded message.

Submission: The senders compose and send their messages to the first mix server. Messages equipped with the signature of the Registration Authority are encrypted with symmetric keys calculated by the senders. Besides the ciphertext a parameter, which is essential for the mix servers to calculate the symmetric keys applied for the decryption, is also transmitted.

Mixing: The first mix server gets the messages from the senders, decrypts and transmits the permutated list of the messages to the next mix server. The messages are being transmitted through the mix network - each server decrypts and permutes the messages - until they arrive to the last mix server, the receiver.

Receiving: The last mix server, the receiver, calculates the symmetric keys and decrypts the messages. The receiver also verifies the eligibility of the sender via the signature of $\mathcal{RA}$.

There are two optional phases. In some systems it is necessary to send a reply to the anonymous sender without revealing his identity and/or to reveal the real identity of the senders.

Anonymous Reply: In this optional phase the receiver can send a message to the anonymous sender by using a value which is calculated and placed into the encrypted message by the sender in the submission phase.

Anonymity Revocation: In this optional phase the sender's ID is revealed. There are two ways for this: with the sender's help or with the joint support of all mix servers.

3.1 Preparation

During preparation the system parameters, public and secret keys are generated involving the Registration Authority and the mix servers.

$\mathcal{RA}$ generates the system parameters, such as: groups G_1, G_2, G_T, generator elements $P \in G_1$, $Q \in G_2$, a bilinear map $e : G_1 \times G_2 \to G_T$, hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : G_T \to \{0,1\}^l$. A key pair is generated for $\beta\beta$, $\overline{r} \in Z_q^*$ and $\overline{r}Q$ denote the secret and public key, respectively. For signing $\mathcal{RA}$ chooses a random secret value $\overline{s} \in Z_q^*$ and outputs public key $(\overline{s}P, \overline{s}Q)$. The secret keys $\overline{r}, \overline{s}$ can be shared in a threshold manner. Public key parameter $\overline{rs}P$ is also calculated with the help of $\beta\beta$.

$\mathcal{M}_\mathbf{1}$	$\mathcal{M}_\mathbf{j}$	$\mathcal{M}_\mathbf{N}$
$SK_{\mathcal{M}_1} = (m_1, x_1)$	$SK_{\mathcal{M}_j} = (m_j, x_j)$	$SK_{\mathcal{R}} = (m_N, x_N)$
$\xrightarrow{m_1 Q}$ $\dots$ $\xrightarrow{(\prod_{k=1}^{j-1} m_k)Q}$	$\xrightarrow{(\prod_{k=1}^{j} m_k)Q}$ $\dots$	$\overline{m}Q = (\prod_{k=1}^{N} m_k)Q$
$\xrightarrow{x_1 m_1 Q}$ $\dots$ $\xrightarrow{(\prod_{k=1}^{j-1} x_k m_k)Q}$	$\xrightarrow{(\prod_{k=1}^{j} x_k m_k)Q}$ $\dots$	$\overline{xm}Q = (\prod_{k=1}^{N} x_k m_k)Q$ $x_N Q$
$PK_{\mathcal{M}_1} = x_1 m_1 Q$	$PK_{\mathcal{M}_j} = x_j(\prod_{k=1}^{j} m_k)Q$	$PK_R = x_N \overline{m}Q$

Table 1: Calculating the server key pairs and the public keys of the mixnet.

Table 1 shows the key generation process of the mix servers. Each mix server $\mathcal{M}_j$ generates a key pair $(SK_{\mathcal{M}_j}, PK_{\mathcal{M}_j})$, where $j = 1, \dots, N$, and the mixnet public keys $\overline{m}Q, \overline{xm}Q \in G_2$. Let $SK_{\mathcal{M}_j} = (m_j, x_j)$, where $m_j, x_j \in Z_q^*$ are random and secret. For calculating the public keys each mix server outputs $(\prod_{k=1}^{j} m_k)Q$ and $(\prod_{k=1}^{j} x_k m_k)Q$ to the next server and $\mathcal{M}_j$ calculates $PK_{\mathcal{M}_j} = x_j(\prod_{k=1}^{j} m_k)Q$. Lastly, R computes $\overline{m}Q$ and $\overline{xm}Q$, where $\overline{m}, \overline{x} \in Z_q^*$. The value $\overline{m}Q$ is used for calculating commitment values μ_i, and $\overline{xm}Q$ is necessary for the encrypted identities ϵ_i, where $\overline{x}$ is never calculated explicitly. R also outputs $x_N Q$ public key for providing anonymous reply.

3.2 Registration

The senders and $\mathcal{RA}$ participate in this phase, $\mathcal{RA}$ verifies senders eligibility and blindly authorizes their messages. Blind BLS-3b signatures ([Chatterjee et al. 2010]) are applied for hiding the messages, hence the Registration Authority is not able link a message with its sender. This means, that even $\mathcal{RA}$ cannot relate IDs to the messages.

$\mathbf{S_i}$		$\mathcal{RA}$		$\beta\beta$
$S_i,\ msg,\ u^{(i)}$		$SK = \overline{s}$		
$H_1(msg) + u^{(i)}P$		$PK = (\overline{s}P, \overline{s}Q, \overline{rs}P)$		
	$\xrightarrow{S_i, H_1(msg)+u^{(i)}P}$			
		$\overline{s}(H_1(msg) + u^{(i)}P)$		
	$\xleftarrow{\overline{s}(H_1(msg)+u^{(i)}P)}$		$\xrightarrow[e(\overline{s}(H_1(msg)+u^{(i)}P), \overline{xm}Q)]{S_i, e(\overline{s}(H_1(msg)+u^{(i)}P), \overline{m}Q)}$	
$u^{(i)}\overline{s}P,\ \overline{s}H_1(msg),\ msg$				$\mu_i = e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{m}Q)^{\overline{r}}$
$e(\overline{s}H_1(msg), Q) \stackrel{?}{=} e(H_1(msg), \overline{s}Q)$				$\epsilon_i = S_i \oplus H_2(e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{xm}Q)^{\overline{r}})$

Table 2: Signature generation.

Table 2 shows all the calculations performed for generating the signature on the message being submitted. Firstly, sender S_i creates the message msg that he would like to send to the receiver R. After calculating the blinded message $H_1(msg) + u^{(i)}P$, where the blinding factor $u^{(i)} \in Z_q^*$ is chosen randomly, S_i sends his ID (S_i) and the blinded message on a *secret, authenticated channel* to $\mathcal{RA}$. Knowing the ID, $\mathcal{RA}$ checks the database containing eligible users' data whether S_i is eligible. If S_i is in the database, $\mathcal{RA}$ blindly signs the message and sends $\overline{s}(H_1(msg) + u^{(i)}P)$ back. Moreover, $\mathcal{RA}$ calculates and transfers values $e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{m}Q), e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{xm}Q)$ and S_i to $\beta\beta$ on an authenticated channel, a commitment value $\mu_i = e(\overline{s}(H_1(msg) + u^{(i)}P, \overline{m}Q)^{\overline{r}}$ for verification purposes and a value $\varepsilon_i = S_i \oplus H_2(e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{xm}Q)^{\overline{r}})$, which is the sender's encrypted ID is calculated by $\beta\beta$. $\beta\beta$ publishes all pairs (μ_i, ε_i) in a permuted order.

After receiving the signed blinded message, S_i is able to obtain a valid signature on msg *i.e.* calculating $\overline{s}H_1(msg)$ with the knowledge of $u^{(i)}\overline{s}P$. S_i is able to verify the signature by checking the equality $e(\overline{s}H_1(msg), Q) = e(H_1(msg), \overline{s}Q)$.

3.3 Message submission and mixing

This phase consists of two parts: submission and mixing. During submission senders calculate the symmetric keys and encrypt their messages. After collecting all submissions the first mix server starts the mixing part. Each mix server decrypts and permutes the messages and transmits them to the next server, the last mix server is the receiver.

During submission each sender composes his plaintext $p = msg||\overline{s}H_1(msg)||a_{s_i}\overline{r}Q$, which consists of the signed message and a parameter $a_{s_i}\overline{r}Q$, where a_{s_i} is a secret, random value. The parameter $a_{s_i}\overline{r}Q$ is optional, it is necessary only for the anonymous reply. S_i chooses $u^{(i)}$ randomly and generates N symmetric keys: $K_j^{(i)} = H_2(e(\overline{rs}P, PK_{\mathcal{M}_j})^{u^{(i)}})$ $(j = 1, \ldots, N)$. S_i uses all keys to encrypt the plaintext. The receiver key $K_R(= K_N)$ is applied first, and the key of the first mix server at last, and $M_1^{(i)} = Enc_{K_1^{(i)}}(Enc_{K_2^{(i)}}(\ldots Enc_{K_R^{(i)}}(p)))$ is obtained. Finally, S_i transfers the pair $(V_1^{(i)}, M_1^{(i)})$ to the first mix, where $V_1^{(i)} = u^{(i)}\overline{r}Q$. The mix servers are able to compute the same symmetric keys with the help of $V_1^{(i)}$.

Only the first mix server collects the pairs from the senders, the other mix servers obtain their input from the previous servers. *Table 3* shows the calculations made by a mix server (denoted by $\mathcal{M}_j$, where $j = 2, ..., N-1$). The first mix server proceeds the same calculations, the only difference is that the input comes from the senders. The computations of the last mix server - the receiver - are a little bit different.

$\mathcal{M}_{\mathbf{j-1}}$	$\mathcal{M}_{\mathbf{j}}$	$\mathcal{M}_{\mathbf{j+1}}$
	$SK_{\mathcal{M}_j} = (m_j, x_j)$	
	$PK_{\mathcal{M}_j} = x_j(\prod_{k=1}^{j} m_k)Q$	
$\xrightarrow{V_j^{(i)}\|\|M_j^{(i)}}$		
	$V_{j+1}^{(i)} = m_j \cdot V_j^{(i)}$	
	$K_j^{(i)} = H_2(e(\overline{s}P, V_{j+1}^{(i)})^{x_j})$	
	$M_{j+1}^{(i)} = Dec_{K_j^{(i)}}(M_j^{(i)})$	
	$\xrightarrow{V_{j+1}^{(i)}\|\|M_{j+1}^{(i)}}$	

Table 3: Calculations of a mix server.

Table 3 summarizes the input, the calculations and the output of a mix server for a message sent by the sender S_i. The mix server $\mathcal{M}_j$ receives n pairs $(V_j^{(i)}||M_j^{(i)})$, where $i = 1, \ldots, n$. Each pair originates from a sender. The mix server uses the first value for calculating the symmetric key that is necessary to decrypt the second value. For each sender, $\mathcal{M}_j$ calculates a randomized symmetric key $K_j^{(i)}$ from values $V_{j+1}^{(i)}$, $\overline{s}P$ with the secret key x_j.

With this key $\mathcal{M}_j$ removes one "encryption layer" from the ciphertexts, hence $M_{j+1}^{(i)} = Enc_{K_{j+1}^{(i)}}(Enc_{K_{j+2}^{(i)}}(\dots Enc_{K_R^{(i)}}(p)))$. $\mathcal{M}_j$ applies a random permutation and sends the permuted list of new pairs to the next mix. The factor m_j assures the unlinkability between $V_j^{(i)}$ and $V_j^{(i+1)}$.

3.4 Receiving the message

The last mix server, the receiver R, obtains the pairs from $\mathcal{M}_{N-1}$. After calculating the symmetric key, R decrypts the message, verifies the eligibility of the anonymous sender, stores the signed message and the parameter that is used to reply anonymously.

R calculates the key $K_R^{(i)} = H_2(e(\overline{s}P, m_N V_N^{(i)})^{x_N})$ from the input values and decrypts $M_N^{(i)}$. After receiving $p = msg||\overline{s}H_1(msg)||a_{s_i}\overline{r}Q$, R checks, whether the message p came from an eligible sender by verifying the signature of $\mathcal{RA}$. R sends $\overline{s}H_1(msg)$ to $\beta\beta$ that publishes $(\overline{s}H_1(msg), \overline{rs}H_1(msg))$. R also computes the commitment value $\mu_i = e(\overline{s}P, m_N V_N^{(i)}) \cdot e(\overline{rs}H_1(msg), \overline{m}Q)$ and verifies its existence on $\beta\beta$. If μ_i is in the database, then $\mathcal{RA}$ has received $H_1(msg) + u^{(i)}P$ during registration, where $u^{(i)}$ is the secret value known only by S_i. Being μ_i and ϵ_i on $\beta\beta$ means, that after the deadline, the sender's identity can be revealed by the mix servers, including R. The receiver stores: $\mu_i||msg||\overline{s}H_1(msg)||a_{s_i}\overline{r}Q$ for eligible senders. The value μ_i is necessary for anonymity revocation and the value $a_{s_i}\overline{r}Q$ is for the reply to the anonymous sender.

3.5 Anonymous reply

In this phase the roles (sender and receiver) are reversed, the receiver R would like to send a message back to the anonymous senders.

The receiver calculates the symmetric key $\widehat{K_R^{(i)}} = H_2(e(\overline{s}P, a_{s_i}\overline{r}Q)^{x_N})$ to encrypt the message $t^{(i)}$. $\widehat{K_R^{(i)}}$ is computed from the public key of $\mathcal{RA}$, the secret key of the receiver and the user's value $a_{s_i}\overline{r}Q$ stored by the receiver. R creates the values $\widehat{V_1^{(i)}} = a_{s_i}\overline{r}Q$ and $\widehat{M_1^{(i)}} = Enc_{\widehat{K_R^{(i)}}}(t^{(i)})$ and sends them to the first mix server. There is a pair for each sender.

$\mathcal{M}_{\mathbf{j-1}}$	$\mathcal{M}_{\mathbf{j}}$	$\mathcal{M}_{\mathbf{j+1}}$
	$SK_{\mathcal{M}_j} = (m_j, x_j)$	
	$PK_{\mathcal{M}_j} = x_j(\prod_{k=1}^{j} m_k)P$	
$\xrightarrow{\widehat{V_j^{(i)}} \| \widehat{M_j^{(i)}}}$		
	$\widehat{V_{j+1}^{(i)}} = m_j \cdot \widehat{V_j^{(i)}}$	
	$\widehat{K_j^{(i)}} = H_2(e(\overline{s}P, \widehat{V_{j+1}^{(i)}})^{x_j})$	
	$\widehat{M_{j+1}}^{(i)} = Enc_{\widehat{K_j^{(i)}}}(\widehat{M_j^{(i)}})$	
	$\xrightarrow{\widehat{V_{j+1}^{(i)}} \| \widehat{M_{j+1}^{(i)}}}$	

Table 4: Calculations of a mix server for the reply messages.

Mix servers perform similar calculations to the mixing phase. The first element is multiplied by m_j, the second value is encrypted. Server $\mathcal{M}_j$ sends the new pairs to the next mix server in permuted order. Server $\mathcal{M}_{N-1}$ outputs all the calculated pairs with index $H_1(\widehat{K_{N-1}^{(i)}})$ on $\beta\beta$.

To obtain the reply plaintext $t^{(i)}$, S_i calculates the symmetric keys ($\widehat{K_j^{(i)}} = H_2(e(\overline{rs}P, PK_{\mathcal{M}_j})^{a_{s_i}})$ $(j = 1, \ldots, N-1)$, $\widehat{K_R^{(i)}} = H_2(e(\overline{sr}P, x_N Q)^{a_{s_i}})$) and looks for the value $H_1(\widehat{K_{N-1}^{(i)}})$ on $\beta\beta$. If the sender finds the hash value, accesses all the corresponding data and decrypts the encrypted message,
$t^{(i)} = Dec_{\widehat{K_R^{(i)}}}(Dec_{\widehat{K_1^{(i)}}}(\ldots Dec_{\widehat{K_{N-1}^{(i)}}}(\widehat{M_N^{(i)}}))).$

3.6 Anonymity revocation

In this optional phase the identity of a sender is determined after a deadline. In general, anonymity revocation should be provided even if the sender is not willing to cooperate with the authority (e.g. an examinee does not want to obtain a bad grade).

For the case when the sender is not cooperative, the mixnet determines the identity of the sender. The real identity of a sender can be retrieved only, if either the sender reveals it himself or all the mix servers together calculate it, as follows. The receiver possessing μ_i calculates $\mu_i^{x_N}$, where x_N is the secret key parameter, and transmits it to the first mix server. Server $\mathcal{M}_j$ powers the received value to the secret key parameter x_j and sends it to the next server. Finally, $\mu_i^{\overline{x}} = [e(\overline{s}P, u^{(i)}\overline{mr}Q) \cdot e(\overline{rs}H_1(msg), \overline{m}Q)]^{\overline{x}}$ is computed. By calculating the hash value $H_2(\mu_i^{\overline{x}})$, which equals to $H_2(e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{rm}Q)^{\overline{x}})$, the identity number S_i is revealed via $S_i = H_2(\mu_i^{\overline{x}}) \oplus \varepsilon_i$, where ε_i is available on $\beta\beta$.

We should mention if the sender is cooperative, the real identity can be revealed without the mix servers in an easier and lower-cost way. The sender provides the secret value $u^{(i)}$ and R can determine the identity of the sender by calculating $H_2(e(\overline{rs}H_1(msg), \overline{xm}Q) \cdot e(u^{(i)}\overline{rs}P, \overline{xm}Q)) \oplus \varepsilon_i$.

From the calculations above one can see the adaption of the property of bilinearity. The mixnet obtains the necessary hash value for anonymity revocation by calculating $H_2(e(\overline{s}(H_1(msg) + u^{(i)}P), \overline{rm}Q)^{\overline{x}})$, $\beta\beta$ needs to compute $H_2(e(\overline{s}(H_1(msg)+u^{(i)}P), \overline{xm}Q)^{\overline{r}})$ and the sender is able to verify the correctness of ε_i on $\beta\beta$ by computing $\epsilon_i = S_i \oplus H_2(e(\overline{rs}H_1(msg), \overline{xm}Q) \cdot e(\overline{rs}P, \overline{xm}Q)^{u^{(i)}})$.

4 Security of BILMIX

4.1 Correctness

First we prove that our scheme is correct concerning the mix process, the anonymous reply and also the process of anonymity revocation.

Definition 8 Correctness. *We call our mixnet correct, if for every plaintext calculated by the receiver there is a corresponding ciphertext in the input list of the mixnet. This means that every plaintext is a multiple decryption of a ciphertext, and no two plaintexts are the multiple decryptions of the same ciphertext.*

The following theorem states that our mixnet is correct.

Theorem 9. *The proposed mix protocol is operating correctly.*

Proof. Each sender S_i sends a pair $(V_1^{(i)} = u^{(i)}\overline{r}Q, M_1^{(i)})$ to the first mix server $\mathcal{M}_1$, where $M_1^{(i)}$ is an N-times encryption of the plaintext p containing the message msg of S_i. $\mathcal{M}_1$ receives n pairs from the senders. $\mathcal{M}_j$ (where $j = 2, \ldots, N-1$) receives a permutation of modified pairs from $\mathcal{M}_{j-1}$. Senders calculate the symmetric keys for all mix servers: $K_j^{(i)} = H_2(e(\overline{rs}P, PK_{\mathcal{M}_j})^{u^{(i)}}) = H_2(e(\overline{rs}P, x_j(\prod_{k=1}^{j} m_k)Q)^{u^{(i)}})$, where $j = 1, \ldots, N-1$. Mix server $\mathcal{M}_J$, $J = 1, \ldots, N-1$ calculates symmetric key $K_J^{(i)} = H_2(e(\overline{s}P, V_{J+1}^{(i)})^{x_J}) = H_2(e(\overline{s}P, m_J V_J^{(i)})^{x_J}) = H_2(e(\overline{s}P, m_J(\prod_{k=1}^{J-1} m_k)u^{(i)}\overline{r}Q)^{x_J}) = H_2(e(\overline{s}P, (\prod_{k=1}^{J} m_k)u^{(i)}\overline{r}Q)^{x_J})$. Because of the bilinear property of pairing e the corresponding keys are the same iff $j = J$.

R receives a set of the pairs $(V_N^{\sigma(i)}, M_N^{\sigma(i)})$ from $\mathcal{M}_{N-1}$, where $\sigma(i)$ is the permutation of $i = 1, \ldots, n$. In order to get the plaintexts the receiver does the following calculations for all $M_N^{(j)}$: $p'_j = Dec_{K_R^{(j)}}(M_N^{(j)}) = Dec_{K_R^{(j)}}(Enc_{K_R^{(i)}}(p_i))$, $j = 1, \ldots, n$, $i = \sigma(j)$ and $K_R^{(j)} = H_2(e(\overline{s}P, m_N V_N^{(j)})^{x_N} = H_2(e(\overline{s}P, m_N(\prod_{k=1}^{N-1} m_k)u^{(j)}\overline{r}Q)^{x_N}) = H_2(e(\overline{s}P, \overline{m}u^{(j)}\overline{r}Q)^{x_N})$. The symmetric key for R calculated by the sender S_i is the following: $K_R^{(i)} = H_2(e(\overline{rs}P, PK_R)^{u^{(i)}}) = H_2(e(\overline{rs}P, x_N\overline{m}Q)^{u^{(i)}})$. Using the bilinear property of pairing e the receiver is able to get a plaintext if and only if $K_R^{(j)} = K_R^{(i)}$, thus the plaintext p'_j is p_i. R calculates μ_i and checks whether it is on $\beta\beta$: $\mu_i = e(\overline{s}P, m_N V_N^{(i)}) \cdot e(\overline{rs}H_1(msg), \overline{m}Q) = e(\overline{s}P, m_N(\prod_{k=1}^{N-1} m_k)u^{(i)}\overline{r}Q) \cdot e(\overline{rs}H_1(msg), \overline{m}Q) = e(\overline{s}P, \overline{m}u^{(i)}\overline{r}Q) \cdot e(\overline{rs}H_1(msg), \overline{m}Q) = e(\overline{s}u^{(i)}P, \overline{rm}Q) \cdot e(\overline{s}H_1(msg), \overline{rm}Q) = e(\overline{s}(u^{(i)} + H_1(msg)P), \overline{rm}Q)$.

The *anonymous reply* works similarly to the message submission. In this case the sender is R and the anonymous receiver is the sender S_i who sent the message msg stored. S_i calculates the value $index = H_1(\widehat{K_{N-1}^{(i)}}) = H_1(H_2(e(\overline{rs}P, PK_{\mathcal{M}_{N-1}})^{a_{s_i}})) = H_1(H_2(e(\overline{rs}P, x_{N-1}(\prod_{k=1}^{N-1} m_k)Q)^{a_{s_i}}))$ and looks for it on $\beta\beta$. Since mix server $\mathcal{M}_{N-1}$ printed the pair $(\widehat{V_N^{(i)}}, \widehat{M_N^{(i)}})$ with the index $H_1(\widehat{K_{N-1}^{(i)}}) = H_1(H_2(e(\overline{s}P, \widehat{V_N^{(i)}})^{x_{N-1}})) = H_1(H_2(e(\overline{s}P, m_{N-1}\widehat{V_{N-1}^{(i)}})^{x_{N-1}}))) = H_1(H_2(e(\overline{s}P, (\prod_{k=1}^{N-1} m_k)a_{s_i}\overline{r}Q)^{x_{N-1}}))$, S_i finds $index$ and two other values: $\widehat{V_N^{(i)}} = (\prod_{k=1}^{N-1} m_k)a_{s_i}\overline{r}Q$ and $\widehat{M_N^{(i)}} = Enc_{\widehat{K_{N-1}^{(i)}}}(\ldots(Enc_{\widehat{K_1^{(i)}}}(Enc_{\widehat{K_R^{(i)}}}(t))))$, where $\widehat{K_j^{(i)}} = H_2(e(\overline{s}P, \widehat{V_{j+1}^{(i)}})^{x_j}) = H_2(e(\overline{s}P, (\prod_{k=1}^{j} m_k)a_{s_i}\overline{r}Q)^{x_j})$ calculated by $\mathcal{M}_j$. Due to the bilinear property of e these keys are the same as S_i calculates for $\mathcal{M}_j$: $H_2(e(\overline{rs}P, PK_{\mathcal{M}_j})^{a_{s_i}}) = H_2(e(\overline{rs}P, x_j(\prod_{k=1}^{j} m_k)Q)^{a_{s_i}})$.

Furthermore, R calculates symmetric key: $\widehat{K_{R_1}} = H_2(e(\overline{s}P, a_{s_i}\overline{r}Q)^{x_N}$ and S_i calculates symmetric key: $\widehat{K_{R_2}} = H_2(e(\overline{rs}P, x_N Q)^{a_{s_i}})$. Pairing e has bilinear property so $\widehat{K_{R_1}} = H_2(e(\overline{s}P, a_{s_i}\overline{r}Q)^{x_N}) = H_2(e(\overline{rs}P, x_N Q)^{a_{s_i}}) = \widehat{K_{R_2}}$ holds.

4.2 Anonymity

4.2.1 Security Model

We consider a *static* adversary in the *semi-honest model*. A model is called semi-honest, if the dishonest users follow the protocol and also keep a record of all intermediate results. An adversary is static, if corrupted players are specified at the beginning of the protocol, they stay corrupted during the whole process and no new ones stand in with them. The adversary observes all public information and possesses all attacked players' secret information (*i.e.* keys, permutation).

4.2.2 The Experiment of Anonymity

We give the definition of anonymity with the help of an experiment. The anonymity property of our system says that an adversary who has access to corrupt players' secret data and observes all the public information of the protocol including views of the Registration Authority and mix servers, input ciphertexts and the shuffled list of output messages, cannot link a message with the sender.

We also assume, that there is at least one mix server and two senders that are not corrupted by the adversary, *i.e.* the secret permutation and secret keys are not revealed to the adversary, furthermore secret keys of the Registration Authority and $\beta\beta$ are not revealed to the adversary. At the end, the adversary tries to give an input/output message pair that originates from a non-corrupted sender according to the knowledge he gained during the registration and the mixing part. The experiment is run by adversary $\mathcal{A}_{anon}$ and the challenger Sys.

Definition 10 Anonymity. The experiment is parameterized by security parameter λ.
1. The challenger Sys generates secret and public keys for input 1^λ. Public keys and public parameters are sent to $\beta\beta$.
2. Secret keys and secret random permutations π_i of the corrupted mix servers are given to $\mathcal{A}_{anon}$.
3. Sys runs the mix process with the list of ciphertexts $(c_1, \ldots, c_n)$ that is published with the output list of plaintexts $(p_{\pi(1)}, \ldots, p_{\pi(n)})$ on $\beta\beta$, where corrupted users' message pairs (p_i, c_i) and all intermediate results are revealed to $\mathcal{A}_{anon}$.
4. The adversary outputs $b' \in \{0, 1\}$ for $(c_0, c_1, p_b, p_{\overline{b}})$ pairs, where $b \in \{0, 1\}$, where plaintexts, ciphertexts are generated by senders that were never corrupted and $\overline{b} = 1 - b$, and assuming that user-specific commitment values and encrypted identity values for all users are listed on $\beta\beta$ in a permuted order.

We define the advantage of the adversary in this experiment by

$$Adv_{Sys,\mathcal{A}_{anon}}(\lambda) = |Pr[b' = b] - \frac{1}{2}|.$$

The mix network process possesses property of **anonymity** if for any PPT adversary $\mathcal{A}_{anon}$ the advantage $Adv_{Sys,\mathcal{A}_{anon}}(\lambda)$ is negligible, where probability is taken over the coin-flips of $\mathcal{A}_{anon}$, as well as random coins used in the experiment for key, permutation, plaintexts and identity number generation.

4.2.3 Proving Anonymity

Let us review the Matching Diffie-Hellman Problem ([Frankel et al. 1996], [Ohkubo and Abe 2000]) and the Matching Find-Guess Problem ([Fujisaki and Okamoto 1999]).

Definition 11 Matching Diffie-Hellman Problem (MDHP) in G_2. For every $r \in Z_q^*$ given $Q, rQ \in G_2$ and $V_0, V_1, rV_b, rV_{\bar{b}} \in G_2$, the problem is to output $b \in \{0, 1\}$.

Definition 12 Matching Find-Guess Problem (MFGP). For every plaintexts x_0, x_1 and for every secret symmetric keys K_0, K_1 given $(Enc_{K_0}(x_0), Enc_{K_1}(x_1), x_b, x_{\overline{b}})$, the problem is to output $b \in \{0,1\}$.

Theorem 13. *If there exists a static adversary in the semi-honest model that breaks sender anonymity in BILMIX, there exists a polynomial time algorithm $\hat{A}$ that solves the co$-BDHP_{1,2}$, or else it solves either the $MFGP$ or the $MDHP$ with probability non-negligibly better than 1/2 in the random oracle model.*

Proof. First of all, we prove that if BILMIX does not provide sender anonymity, *i.e. there exists a polynomial time algorithm $\mathcal{A}_{anon}$*, we can construct an efficient algorithm $\mathcal{A}$ that solves the following problems with the help of $\mathcal{A}_{anon}$.

Definition 14 BILMIX Problem (BP). *For any $P, \overline{s}P \in G_1$ and $Q, rQ, rxQ \in G_2$ there are tuples $(V_0, M_\xi^{(0)})$, $(V_1, M_\xi^{(1)})$, $(rV_b, M_{\xi+1}^{(b)})$, $(rV_{\overline{b}}, M_{\xi+1}^{(\overline{b})})$ given, where $\overline{s}, r, x \in Z_q^*$, $V_i \in G_2$, moreover $M_j^{(i)}$ are ciphertexts for $i \in \{0,1\}$ and $j \in \{\xi, \xi+1\}$, where $M_{\xi+1}^{(b)} = Dec_{K_\xi^{(b)}}(M_\xi^{(b)})$ and $K_\xi^{(b)} = H_2(e(\overline{s}P, V_b)^{rx})$. The problem is to output $b \in \{0,1\}$.*

We construct the efficient algorithm $\mathcal{A}$, as follows. $\mathcal{A}$ simulates the operation of the players of *Sys*, *i.e.* mix servers, $\mathcal{RA}$ and the corrupt senders. The honest senders and the honest mix server, denoted by $(S_0, S_1), \mathcal{M}_\xi$, respectively, are chosen at the beginning. $\mathcal{A}$ sets rQ to be the message being sent to the next mix server by $\mathcal{M}_\xi$ and rxQ to be the public key of $\mathcal{M}_\xi$. Afterwards, $\mathcal{A}$ generates the remaining secret/public keys and input messages. $\mathcal{A}$ during the simulation rejects secret key exposure queries with respect ξth key. $\mathcal{A}$ simulates the view by simulating the keys and the list of ciphertexts in the following way.

$\mathcal{A}$ sets $\overline{s}P$ as a public key for $\mathcal{RA}$ and simulates keys of the ascending servers by randomly choosing $m_{\xi+1}, m_{\xi+2}, \ldots, m_N$ and $x_{\xi+1}, x_{\xi+2}, \ldots, x_N$ to be the secret keys, and generating $m_{\xi+1}rQ$, $m_{\xi+1}m_{\xi+2}rQ, \ldots, (\prod_{k=\xi+1}^{N} m_k)rQ$ and $x_{\xi+1}m_{\xi+1}rQ$, $x_{\xi+2}m_{\xi+1}m_{\xi+2}rQ, \ldots, x_N(\prod_{k=\xi+1}^{N} m_k)rQ$ to be the public keys. For the descending servers $\mathcal{A}$ randomly chooses $m_{\xi-1}, m_{\xi-2}, \ldots, m_1$ and $x_{\xi-1}, x_{\xi-2}, \ldots, x_1$ as secret keys, and sets $Q, m_{\xi-1}^{-1}Q$, $m_{\xi-1}^{-1}m_{\xi-2}^{-1}Q, \ldots, (\prod_{k=2}^{\xi-1} m_k^{-1})Q$ and $x_{\xi-1}Q, x_{\xi-2}m_{\xi-1}^{-1}Q$, $x_{\xi-3}m_{\xi-1}^{-1}m_{\xi-2}^{-1}Q, \ldots$, $x_1(\prod_{k=2}^{\xi-1} m_k^{-1})Q$ as public keys. The following public values are also generated for all servers (starting from the first one): $x_1(\prod_{k=2}^{\xi-1} m_k^{-1})Q, \ldots$, $(\prod_{k=1}^{\xi-2} x_k)m_{\xi-1}^{-1}Q$, $(\prod_{k=1}^{\xi-1} x_k)Q$, $(\prod_{k=1}^{\xi-1} x_k)xrQ$, $(\prod_{k=1}^{\xi-1} x_k)xx_{\xi+1}rm_{\xi+1}Q, \ldots$, $(\prod_{k=1}^{\xi-1} x_k)x(\prod_{k=\xi+1}^{N} x_k m_k)rQ$.

$\mathcal{A}$ reveals the secret values of corrupt users and all public ones to $\mathcal{A}_{anon}$, then randomly chooses permutations for the corrupt servers and also reveals them to $\mathcal{A}_{anon}$. For simplicity, we set all corrupt server permutations to the identity.

The list of messages are simulated as follows. Denote the elements of the list sent to $\mathcal{M}_j$ by $V_j^{(i)}||M_j^{(i)}$, where $i = 1, \ldots, n$ is the index number of a sender and $j = 1, \ldots, N$ the sequential number of a mix server. $\mathcal{A}$ simulates the list for $\mathcal{M}_{\xi+1}$ by inserting $(rV_b, M_{\xi+1}^{(b)})$ and $(rV_{\bar{b}}, M_{\xi+1}^{(\bar{b})})$ into random positions and for the rest of the list $\mathcal{A}$ randomly chooses $y_{\xi+1}^{(i)} \in Z_q^*$, calculates $V_{\xi+1}^{(i)} = y_{\xi+1}^{(i)}Q$ and randomly chooses $M_{\xi+1}^{(i)}$ from the ciphertext space. Then $\mathcal{A}$ simulates the list for $\mathcal{M}_\xi$, he inserts $(V_0, M_\xi^{(0)})$ and $(V_1, M_\xi^{(1)})$ into the proper positions, for the rest of the list randomly chooses $y_\xi^{(i)} \in Z_q^*$ and sets $V_\xi^{(i)} = y_\xi^{(i)}Q$. $\mathcal{A}$ computes $M_\xi^{(i)} = Enc_{K_\xi^{(i)}}(M_{\xi+1}^{(i)})$, where $K_\xi^{(i)} = H_2(e(\overline{sr}P, rxQ)^{u^{(i)}})$, where $u^{(i)} = m_1^{-1} \cdot \ldots \cdot m_{\xi-1}^{-1} y_\xi^{(i)} \bar{t}^{-1}$, where $t \in Z_q^*$ is randomly chosen.

For descending servers $\mathcal{A}$ randomly chooses $m_j^{(i)}$ for $j = \xi-1, \ldots, 1$, calculates $V_j^{(i)} = (m_j^{(i)})^{-1} \cdot V_{j+1}^{(i)}$, and $K_j^{(i)} = H_2(e(\bar{s}P, V_{j+1}^{(i)})^{x_j})$ and computes $M_j^{(i)} = Enc_{K_j^{(i)}}(M_{j+1}^{(i)})$.

For ascending servers $\mathcal{A}$ randomly chooses $m_j^{(i)}$ for $j = \xi + 2, \ldots, N - 1$, calculates $V_j^{(i)} = m_{j-1}^{(i)} \cdot V_{j-1}^{(i)}$ and $K_j^{(i)} = H_2(e(\bar{s}P, m_j^{(i)}V_j^{(i)})^{x_j})$ and computes $M_j^{(i)} = Dec_{K_j^{(i)}}(M_{j-1}^{(i)})$.

After decryption the last server receives the plaintexts that are revealed to $\mathcal{A}_{anon}$. Plaintext p_i is a form of $msg_i||sign_i||rv_i$. $\mathcal{A}$ outputs μ_i, ε_i, where $\mu_i = e(\bar{s}P, m_N V_N^{(i)}) \cdot e(\bar{t}sign_i, \prod_{k=\xi+1}^{N} m_k rQ)$ and chooses $\varepsilon_i \in \{0,1\}^l$ randomly.

During simulation $\mathcal{A}$ calls random oracles for calculating hash values. After $\mathcal{A}$ reveals the list of plaintexts, ciphertexts and all intermediate results, *i.e.* properly simulated views and lists, $\mathcal{A}_{anon}$ distinguishes the two messages in $\mathcal{M}_N$'s list originate from S_0 and S_1, *i.e* outputs b. From this result $\mathcal{A}$ can derive the correspondence between the pairs given in *BILMIX Problem* and output b. We get the advantage of static adversary as $Adv_{\mathcal{A}} = Adv_{\mathcal{A}_{anon}}$.
As a second step we prove the following lemma.

Lemma 15. *If $\mathcal{A}$ breaks BP, there exists a polynomial time algorithm $\hat{\mathcal{A}}$ that breaks the $co-BDHP_{1,2}$, or else it breaks either the $MFGP$ or the $MDHP$.*

For solving the $co-BDHP_{1,2}$ or else for solving either the $MFGP$ or the $MDHP$ we create algorithms $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$, $\hat{\mathcal{A}}_{MFGP}$, $\hat{\mathcal{A}}_{MDHP}$, respectively. We construct $\hat{\mathcal{A}}$ as follows.

1. $\hat{\mathcal{A}}$ receives a $co-BDHP_{1,2}$ instance $(\mathbf{P}, a\mathbf{P}, b\mathbf{P}) \in G_1^3$ and $(\mathbf{Q}, a\mathbf{Q}, c\mathbf{Q}) \in G_2^3$ and an MFGP instance $(Enc_{K_0}(x_0), Enc_{K_1}(x_1), x_b, x_{\bar{b}})$ and an MDHP instance $(Q, rQ, (V_0, V_1), (rV_b, rV_{\bar{b}}))$.

2. Input each instance to the appropriate algorithm, to $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$ or $\hat{\mathcal{A}}_{MFGP}$ or $\hat{\mathcal{A}}_{MDHP}$.

3. Output the data received from the algorithms as a solution to the input instances.

Let list the steps of the algorithms $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$, $\hat{\mathcal{A}}_{MFGP}$ and $\hat{\mathcal{A}}_{MDHP}$. $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$ after receiving its input generates a problem instance for the subroutine $\mathcal{A}$. $\mathcal{A}$ solves the problem and outputs the necessary data to $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$. $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$ passes the output to $\hat{\mathcal{A}}$. $\mathcal{A}$ has access to the random oracle H_2. Denote the maximum number of queries to H_2 by $qnum$, that is polynomial in the security parameter λ. We assume that there is $j \in \{0,1\}$, such that the value $e(\overline{s}P, V_j)^{rx}$ is among the values that $\mathcal{A}$ can ask from H_2. If $\mathcal{A}$ asks the proper $e(\overline{s}P, V_j)^{rx}$, $\mathcal{A}$ solves the $co-BDHP_{1,2}$. Let Q_i denote the value that is asked from H_2. We denote the probability that $e(\overline{s}P, V_j)^{rx}$ is element of the list:

$$P_{co-BDHP_{1,2}} := Pr[\exists i \in \{1,\ldots,qnum\}, \exists j \in \{0,1\} : Q_i = e(\overline{s}P, V_j)^{rx}].$$

Algorithm $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$:
1. Receives $(\mathbf{P}, a\mathbf{P}, b\mathbf{P}) \in G_1^3$ and $(\mathbf{Q}, a\mathbf{Q}, c\mathbf{Q}) \in G_2^3$.
2. Sets $rQ := \mathbf{Q}, rxQ := c\mathbf{Q}, Q := r^{-1}\mathbf{Q}$, where $r \in Z_q^*$ randomly chosen.
3. Sets $P := \mathbf{P}, \overline{s}P := b\mathbf{P}$.
4. Chooses $b_0 \in \{0,1\}$ randomly.
5. Sets $V'_{b_0} := a\mathbf{Q}$ and randomly chooses $V'_{\overline{b_0}} \in G_2$.
6. Calculates $V_0 := r^{-1}V'_0$ and $V_1 := r^{-1}V'_1$.
7. Randomly chooses ciphertexts $M_\xi^{(0)}, M_\xi^{(1)}$ from the ciphertext space, keys K_0, K_1 from the keyspace and calculates $M_{\xi+1}^{(i)} = Dec_{K_i}(M_\xi^{(i)})$ for $i \in \{0,1\}$.
8. Chooses $b \in \{0,1\}$ randomly.
9. Chooses $i \in [1,\ldots,qnum]$ randomly.
10. Sends problem instance $\{P, \overline{s}P, Q, rQ, rxQ, (V_0, M_\xi^{(0)}),(V_1, M_\xi^{(1)}),$
$(V'_b, M_{\xi+1}^{(b)}),\ (V'_{\overline{b}}, M_{\xi+1}^{(\overline{b})})\}$ to $\mathcal{A}$.
11. $\mathcal{A}$ makes a query to H_2. If he asks the i-th query, output the value $\mathcal{A}$ asked and stop.

We note that the simulation is perfect only if the proper Q_i – that gives the solution to the problem instance – is asked, otherwise no query is sent to H_2. In case $\hat{\mathcal{A}}$ receives an MFGP problem instance, it is forwarded to $\hat{\mathcal{A}}_{MFGP}$.

Algorithm $\hat{\mathcal{A}}_{MFGP}$:
1. Receives $(M_\xi^{(0)} := Enc_{K_0}(x_0), M_\xi^{(1)} := Enc_{K_1}(x_1), M_{\xi+1}^{(b)} := x_b, M_{\xi+1}^{(\overline{b})} := x_{\overline{b}})$.
2. Randomly chooses $P \in G_1$, $Q \in G_2$ and $\overline{s}, r, x \in Z_q^*$ calculates $\overline{s}P$,rQ and rxQ.
3. Randomly chooses $V_0, V_1 \in G_2$ and calculates $V'_0 = rV_0$, $V'_1 = rV_1$.
4. Randomly chooses $b_0 \in \{0,1\}$.
5. Sends problem instance $\{P, \overline{s}P, Q, rQ, rxQ, (V_0, M_\xi^{(0)}),(V_1, M_\xi^{(1)}),$

$(V'_{b_0}, M^{(b)}_{\xi+1})$, $(V'_{\bar{b}_0}, M^{(\bar{b})}_{\xi+1})\}$ to $\mathcal{A}$.
6. $\mathcal{A}$ makes a query to H_2. For all queries randomly chooses a value form the keyspace.
7. Returns value b that is output by $\mathcal{A}$.

If $\hat{\mathcal{A}}$ receives a MDHP problem instance, then it is forwarded to $\hat{\mathcal{A}}_{MDHP}$.

Algorithm $\hat{\mathcal{A}}_{MDHP}$:
1. Receives $(Q, rQ, (V_0, V_1), (rV_b, rV_{\bar{b}}))$.
2. Randomly chooses $P \in G_1$, $\overline{s}, x \in Z_q^*$ and calculates $\overline{s}P$ and rxQ.
3. Randomly chooses ciphertexts $M^{(0)}_{\xi}, M^{(1)}_{\xi}$ from the ciphertext space, keys K_0, K_1 from the keyspace and calculates $M^{(i)}_{\xi+1} = Dec_{K_i}(M^{(i)}_{\xi})$ for $i \in \{0, 1\}$.
4. Randomly chooses $b_0 \in \{0, 1\}$.
5. Sends problem instance $\{P, \overline{s}P, Q, rQ, rxQ, (V_0, M^{(0)}_{\xi}), (V_1, M^{(1)}_{\xi}), (rV_b, M^{(b_0)}_{\xi+1}),$ $(rV_{\bar{b}}, M^{(\overline{b_0})}_{\xi+1})\}$ to $\mathcal{A}$.
6. $\mathcal{A}$ makes a query to H_2. For all queries randomly chooses a value form the keyspace.
7. Returns value b that is output by $\mathcal{A}$.

$\hat{\mathcal{A}}_{MFGP}$ generates the instance $\{P, \overline{s}P, Q, rQ, rxQ, (V_0, M^{(0)}_{\xi}),$ $(V_1, M^{(1)}_{\xi}), (V'_{b_0}, M^{(b)}_{\xi+1})$ $(V'_{\bar{b}_0}, M^{(\bar{b})}_{\xi+1})\}$ and $\hat{\mathcal{A}}_{MDHP}$ calculates the instance $\{P, \overline{s}P, Q, rQ, rxQ, (V_0, M^{(0)}_{\xi}), (V_1, M^{(1)}_{\xi}), (rV_b, M^{(b_0)}_{\xi+1}),$ $(rV_{\bar{b}}, M^{(\overline{b_0})}_{\xi+1})\}$.
Note that these instances might not be correct to $\mathcal{A}$, since b may not equal to b_0. If $b \neq b_0$, $\mathcal{A}$ does not stop in t_{poly} steps, where t_{poly} is polynomial in λ. Therefore $\hat{\mathcal{A}}$ chooses $b \in \{0, 1\}$ randomly.

The success probability of $\hat{\mathcal{A}}$ is calculated as follows. Assume that $P_{co-BDHP_{1,2}}$ is not negligible. The probability that $\hat{\mathcal{A}}_{co-BDHP_{1,2}}$ outputs $Q_i = e(\overline{s}P, V_{b_0})^{rx}$ is $\frac{P_{co-BDHP_{1,2}}}{2qnum}$. That is not negligible.

The other case is when $P_{co-BDHP_{1,2}}$ is negligible. $\hat{\mathcal{A}}_{MFGP}$ and $\hat{\mathcal{A}}_{MDHP}$ receive an input and generate the problem instances to $\mathcal{A}$. The probability that $b = b_0$ and $b \neq b_0$ is $\frac{1}{2}$. In case $b = b_0$ the success probability equals to the success probability of $\mathcal{A}$, denoted by $P_{\mathcal{A}}$. When $b \neq b_0$, $\hat{\mathcal{A}}$ chooses $b \in \{0, 1\}$, let P_b denote the probability that b is chosen. Hence, the probability that $\hat{\mathcal{A}}$ outputs the bit b is $(1 - P_{co-BDHP_{1,2}})(\frac{1}{2}P_{\mathcal{A}} + \frac{1}{2}P_b)$. Assuming $P_{\mathcal{A}} \geq \frac{1}{2} + \mu$, where μ is not negligible and $P_b = \frac{1}{2}$ we get

$$(1 - P_{co-BDHP_{1,2}})(\frac{1}{2}P_{\mathcal{A}} + \frac{1}{2}P_b) \geq (\frac{1}{2} + \frac{\mu}{2}) - (\frac{1}{2} + \frac{\mu}{2})P_{co-BDHP_{1,2}} \geq \frac{1}{2} + \hat{\mu},$$

for some $\hat{\mu}$ that is not negligible. Therefore the success probability of $\hat{\mathcal{A}}$ is not negligible. Since all algorithms are efficient, $\hat{\mathcal{A}}$ is efficient as well.

We also prove that if $DDH_{2,2,2}$ Problem is hard, then the adversary is not able to link the plaintexts with their senders, even if $\mathcal{RA}$ reveals all data from the database, except the secret key $\overline{s}$ and $\mathcal{R}$ provides all data including the secret keys. We define variations of a new problem called *Divisible Decisional Factorized Diffie-Hellman Problem*(DDF-DHP) as follows.

Definition 16 The $DDF-DH_{i,j}$ Problem. Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ and values $i, j \in \{1, 2\}$ we define the $DDF-DH_{i,j}$ Problem to be the following: Given aP_j, bP_j, xP_j and cP_i, with $a, b, c, x \in Z_q^*$, we are asked to decide whether $c \equiv x/ab \pmod q$.

Lemma 17. *Given a pairing problem instance $\Gamma = (q, G_1, G_2, G_T, P_1, P_2, e)$ if there is an efficient adversary that breaks $DDF-DH_{i,j}$ Problem, then we can construct a polynomial time algorithm that breaks the $DDH_{j,j,j}$ Problem, where values $i, j \in \{1, 2\}$.*

Proof. $\mathcal{A}_{DDH_{j,j,j}}$ algorithm is constructed as follows. $\mathcal{A}_{DDH_{j,j,j}}$ receives input $\overline{a}P_j$, $\overline{b}P_j$, $\overline{c}P_j$, with $\overline{a}, \overline{b}, \overline{c} \in Z_q^*$, generates and submits input instance $(aP_j := \overline{a}P_j, bP_j := \overline{b}P_j, xP_j := d\overline{c}P_j, cP_i := dP_i)$ to $\mathcal{A}_{DDF-DH_{i,j}}$, where $d \in Z_q^*$ chosen randomly. $\mathcal{A}_{DDH_{j,j,j}}$ returns the output (true or false) received from $\mathcal{A}_{DDF-DH_{i,j}}$. If $\mathcal{A}_{DDF-DH_{i,j}}$ is efficient, then $\mathcal{A}_{DDH_{j,j,j}}$ is efficient as well.

Theorem 18. *If there exists a static, efficient adversary in the semi-honest model that links plaintexts with their senders in BILMIX, there exists a polynomial time algorithm $\mathcal{A}_{DDF-DH_{1,2}}$ that solves $DDF-DH_{1,2}$ Problem in the random oracle model.*

Proof. $\mathcal{R}A$ possesses a list of triplets $(S_i, H_1(msg) + u^{(i)}P, \overline{s}(H_1(msg) + u^{(i)}P))$ and $\mathcal{R}$ has a list of tuples $(\overline{msg}, H_1(\overline{msg}), \overline{s}H_1(\overline{msg}), \overline{rs}H_1(\overline{msg}), \overline{rm}u^{(i)}Q)$. Let $\mathcal{A}'$ denote the adversary who can link plaintexts with their senders, *i.e.* decide whether an element $\overline{msg}$ from the list of $\mathcal{R}$ is generated by S_i from the list of $\mathcal{R}A$. We construct the efficient algorithm $\mathcal{A}_{DDF-DH_{1,2}}$ as follows.

Algorithm $\mathcal{A}_{DDF-DH_{1,2}}$:

1. Receives $(\mathbf{P}, \mathbf{Q}, a\mathbf{Q}, b\mathbf{Q}, x\mathbf{Q}, c\mathbf{P})$, where $\mathbf{P}$ and $\mathbf{Q}$ are generators of G_1 and G_2, respectively.
2. Simulate public key inputs for $\mathcal{A}'$: $P := \mathbf{P}$, $\overline{s}P := t\mathbf{P}$, $\overline{sr}P := v\mathbf{P}$, where $t, v \in Z_q^*$ are randomly chosen
$Q := \mathbf{Q}$, $\overline{s}Q := t\mathbf{Q}$, $\overline{r}Q := a\mathbf{Q}$, $\overline{m}Q := b\mathbf{Q}$, $\overline{xm}Q := lb\mathbf{Q}$, where $l \in Z_q^*$ is randomly chosen
3. Simulate a list element of $\mathcal{R}$: $\overline{msg} := bs$, $H_1(\overline{msg}) := T$, $\overline{s}H_1(\overline{msg}) := tT$, $\overline{rs}H_1(\overline{msg}) := M$, $\overline{rm}u^{(i)}Q := x\mathbf{Q}$, where bs is a random bitsring and $T, M \in G_1$ are chosen randomly
4. Simulate a list element of $\mathcal{RA}$: $S_i := I$, $H_1(msg) + u^{(i)}P := T + c\mathbf{P}$,

$\overline{s}(H_1(msg) + u^{(i)}P) := t(T + c\mathbf{P})$, where I is a random bitstring
5. Simulates a list element of $\beta\beta$: $\mu_i := e(t\mathbf{P}, x\mathbf{Q}) \cdot e(M, b\mathbf{Q}), \varepsilon_i := I \oplus H_2(\mu_i^l)$
6. Sends all data to $\mathcal{A}'$.
7. Returns value $b \in \{true, false\}$ that is output by $\mathcal{A}'$.

During simulation $\mathcal{A}_{DDF-DH_{1,2}}$ calls random oracles for calculating hash values. We get the advantage of static adversary as $Adv_{\mathcal{A}_{DDF-DH_{1,2}}} = Adv_{\mathcal{A}'}$.

5 Conclusions and future work

As far as we know, our proposal (BILMIX) is the first hybrid mixnet based on *asymmetric bilinear pairings.* We have given an experiment-based security definition of sender anonymity and also proved that the mixnet we proposed provides anonymity against static adversaries in the semi-honest model, assuming that the co-Bilinear Diffie-Hellman Problem, the Matching Find-Guess Problem and the Matching Diffie-Hellman Problem are hard. We also defined variations of a new problem called Divisible Decisional Factorized Diffie-Hellman Problem (DDF-DHP), we show that finding connection between data stored by $\mathcal{RA}$ and $\mathcal{R}$ is at least as hard as breaking DDF-DHP, with the assumption that secret keys of $\mathcal{RA}$ and $\beta\beta$ are kept secret. The next step is to extend BILMIX to achieve end-to-end verifiability in a malicious model.

Acknowledgment

The authors thank to the anonymous reviewer for the valuable remarks. This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was co-financed by the Hungarian Government and the European Social Fund. The first author is supported by the Hungarian National Foundation for Scientific Research Grant No. NK 104208.

References

[Backes et al. 2012] Backes, M., Goldberg I., Kate, A., Mohammadi, E.:"Provably Secure and Practical Onion Routing"; 2012 IEEE 25th Computer Security Foundations Symposium, Cambridge, MA, (2012), 369–385.

[Boldyreva 2003] Boldyreva, A.: "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme"; International Workshop on Theory and Practice in Public Key Cryptography (PKC) 2003 Proceedings, VOL 2567 of LNCS, (2003), 31–46.

[Boneh and Franklin 2001] Boneh, D., Franklin, M.: "Identity based encryption from the Weil pairing"; Advances in Cryptography - Proceedings of Crypto 2001, VOL 2139 of LNCS, (2001), 213–229.

[Boneh et al. 2002] Boneh, D., Lynn, B., Shacham, H.: "Short signatures from the Weil pairing."; In Asiacrypt01, volume 2248 of LNCS, pages 514532. Springer, 2002.

[Camenisch and Lysyanskaya 2001] Camenisch, J., Lysyanskaya, A.: "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation"; In: Pfitzmann B. (eds) Advances in Cryptology EUROCRYPT 2001. EUROCRYPT 2001. Lecture Notes in Computer Science, vol 2045. Springer, Berlin, Heidelberg, (2001) 93–118.

[Chatterjee et al. 2010] Chatterjee, S., Hankerson, D., Knapp, E., Menezes, A.: "Comparing two pairing-based aggregate signature schemes."; Designs, Codes and Cryptography, VOL 55
2-3,(2010), 141–167.

[Chaum 1981] Chaum, D.: "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms"; Commun. ACM 24(2), (1981), 84–88.

[Chaum et al. 2017] Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., De Ruiter, J., Sherman, T. A.: "cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations"; International Conference on Applied Cryptography and Network Security, ACNS 2017, VOL 10355 of LNCS, (2017), 557–578.

[Chen et al. 2011] Chen, Y., Chou, J., Sun, H., Cho, M.: "A novel electronic cash system with trustee-based anonymity revocation from pairing"; Electronic Commerce Research and Applications, Volume 10, Issue 6, NovemberDecember 2011, Elsevier, (2011) 673–682.

[Chen et al. 2015] Chen, C., Asoni, D. E., Barrera, D., Danezis, G., Perrig, A.: "HORNET: high-speed onion routing at the network layer"; In Proc. 22nd ACM Conference on Computer and Communications Security (2015), 1441–1454.

[Danezis et al. 2003] Danezis, G., Dingledine, R., Mathewson, N.: "Mixminion: design of a type III anonymous remailer protocol"; in: IEEE Symposium on Security and Privacy, (2003), 2–15.

[Danezis et al. 2009] Danezis, G., Diaz, C., Syverson, P.: "Systems for Anonymous Communications"; Handbook of Financial Cryptography and Security, Cryptography and Network Security Series, (2009), 341–389.

[Danezis and Serjantov 2004] Danezis, G. Serjantov, A.: "Statistical Disclosure or Intersection Attacks on Anonymity Systems"; J. Fridrich (Ed.): IH 2004, LNCS 3200, (2004), 293–308.

[Erdin et al. 2015] Erdin, E., Zachor, C., Gunes, M. H.: "How to Find Hidden Users: A Survey of Attacks on Anonymity Networks"; IEEE Communications Surveys and Tutorials, vol. 17, no. 4, (Fourthquarter 2015), 2296–2316.

[Federrath et al. 2006] Köpsell, S., Wendolsky, R., Federrath, H.: "Revocable Anonymity." In: Müller G. (eds) Emerging Trends in Information and Communication Security. Lecture Notes in Computer Science, vol 3995. Springer, Berlin, Heidelberg (2006), 206–220.

[Frankel et al. 1996] Frankel, Y., Tsiounis, Y., Yung, M.: ""Indirect Discourse Proofs": Achieving Efficient Fair Off-Line E-Cash"; In: Kim K., Matsumoto T. (eds) Advances in Cryptology ASIACRYPT '96. ASIACRYPT 1996. Lecture Notes in Computer Science, VOL 1163. (1996), 286–300.

[Fujisaki and Okamoto 1999] Fujisaki, E., Okamoto, T.: "Secure integration of asymmetric and symmetric encryption schemes."; Advances in Cryptography - Proceedings of Crypto 1999, VOL 1666 of LNCS, (1999), 537–554.

[Galbraith et al. 2008] Galbraith, S., Paterson, K., Smart, N.: "Pairings for cryptographers", Discrete Applied Mathematics, VOL 156, (2008), 3113–3121.

[Golle et al. 2004] Golle, P., Jakobsson, M., Juels, A., Syverson, P.: "Universal reencryption for mixnets."; In Proceedings of the 2004 CT-RSA Conference, volume 2964, Springer-Verlag, (2004), 163-178.

[Hankerson et al. 2008] Hankerson, D., Menezes, A., Scott, M.: "Software implementation of pairings.", in: M. Joye, G. Neven (Eds.), Identity-Based Cryptography, IOS Press, (2008), 188-206.

[Huang et al. 2006] Huang, L., Yamane, H., Matsuura, K., Sezaki, K.: "Silent cascade: Enhancing location privacy without communication qos degradation."; In Clark, J.

A., Paige, R. F., Polack, F., and Brooke, P. J., editors, SPC, volume 3934 of Lecture Notes in Computer Science, Springer, (2006), 165-180.

[Huszti and Kovacs 2015] Huszti, A., Kovacs, Z.: “Bilinear Pairing-based Hybrid Mixnet with Anonymity Revocation”; 1th ICISSP, Angers, France, 9-11 February, (2015), 238–245.

[Jakobsson et al. 2002] Jakobsson, M., Juels, A., Rivest, R. L.: “Making mix nets robust for electronic voting by randomized partial checking.”; In Proceedings of the 11th USENIX Security Symposium,Berkeley, CA, USA. USENIX Association, (2002), 339-353.

[Joux 2004] Joux, A.: “A one round protocol for tripartite Diffie-Hellman.”; Algorithmic Number Theory: 4th International Symposium, ANTS-IV, VOL 1838 of LNCS (2000), 385–393. Full version: Journal of Cryptology, 17, (2004), 263–276.

[Kate et al 2007] Kate, A., Zaverucha, G., Goldberg, I.: ”Pairing-Based Onion Routing”; Privacy Enhancing Technologies: 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, Springer Berlin Heidelberg, (2007), 95–112

[Kwon et al. 2016] Kwon, A., Lazar, D., Devadas, S., Ford, B.: ”Riffle An Efficient Communication System With Strong Anonymity”; Proc. on Privacy Enhancing Technologies, (2), (2016), 115–134.

[Menezes 1993] Menezes, A.: ”An introduction to Pairing-Based Cryptography”; Contemporary Mathematics, Vol. 477, (2009), 47–65.

[Ohkubo and Abe 2000] Ohkubo, M., Abe, M.: “A Length-Invariant Hybrid Mix”; ASIACRYPT2000, T. Okamoto (Ed.) Springer-Verlag Berlin Heidelberg, LNCS 1976, (2000), 178–191.

[Pfitzmann and Waidner 1985] Pfitzmann, A., Waidner, M.: ”Networks without user observability design options”; In: Proc. Advances in cryptologyEUROCRYPT 85. (1986), 245–253.

[Preneel et al. 2003] Claessens, J., Díaz, C., Goemans, C., Preneel, B., Vandewalle, J., Dumortier, J.: ”Revocable anonymous access to the Internet”, Internet Research, Vol. 13 Issue: 4, (2003), 242–258.

[Raymond 2001] Raymond, JF.: ”Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems”; Federrath H. (eds) Designing Privacy Enhancing Technologies. Lecture Notes in Computer Science, vol 2009. Springer, Berlin, Heidelberg, (2001), 10–29.

[Ren and Wu 2010] Ren, J., Wu, J.: “Survey on anonymous communications in computer networks”; Elsevier, Computer Communications, Volume 33, Issue 4, (2010), 420–431.

[Sampigethaya and Poovendran 2006] Sampigethaya, K., Poovendran, R.: “A Survey on Mix Networks and Their Secure Applications”; Proceedings of the IEEE, Volume 94, Issue 12, (2006), 2142–2181.

[Smart and Vercauteren 2005] Smart, N.P., Vercauteren, F.:“On Computable Isomorphisms in Efficient Pairing Based Systems”; Cryptology ePrint Archive, Report 2005/116.2005

[Suriadi et al. 2008] Suriadi, S., Foo, E., Smith, J.: ”A User-Centric Protocol for Conditional Anonymity Revocation”; Trust, Privacy and Security in Digital Business: 5th International Conference, TrustBus 2008 Turin, Italy, September 4-5, 2008 Proceedings, Springer Berlin Heidelberg, (2008) 185–194.

[Tor project 2003] The Tor project, https://www.torproject.org, (2003), accessed Nov 2017.

[Wright et al. 2003] Wright, M., Adler, M., Levine, B.N., Shields, C.: ”Defending anonymous communication against passive logging attacks”; Proc. IEEE Symposium on Research in Security and Privacy, Berkeley, CA, (2003), 28–41.

[Zhong 2009] Zhong, S.: “Identity-based mix: Anonymous communications without public key certificates”; Computers and Electrical Engineering 35(5), (2009), 705–711.