Proxy Cryptography Revisited

Anca Ivan, Yevgeniy Dodis
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, New York, NY 10012
{*ivan,dodis*}*@cs.nyu.edu*

Abstract

In this work we revisit and formally study the notion of proxy cryptography. Intuitively, various proxy functions allow two cooperating parties F (the "FBI") and P (the "proxy") to duplicate the functionality available to the third party U (the "user"), without being able to perform this functionality on their own (without cooperation). The concept is closely related to the notion of threshold cryptography, except we deal with only two parties P and F, and place very strict restrictions on the way the operations are performed (which is done for the sake of efficiency, usability and scalability). For example, for decryption (resp. signature) P (F) sends a single message to F (P), after which the latter can decrypt (sign) the message. Our formal modeling of proxy cryptography significantly generalizes, simplifies and simultaneously clarifies the model of "atomic proxy" suggested by Blaze and Strauss [4]. In particular, we define bidirectional and unidirectional variants of our model[1], and show extremely simple generic solutions for proxy signature and encryption in these models. We also give more efficient solutions for several specific schemes. We conclude that proxy cryptography is a relatively simple concept to satisfy when looked from the correct and formal standpoint.

1 Introduction

The Blaze and Strauss [4] paper introduced the notion of (atomic) proxy cryptography. The authors define "atomic proxy functions" as functions that transform ciphertext corresponding to one key into ciphertext for another key without revealing any information about the secret decryption keys or the clear text. In the case of signatures, the proxy functions convert a valid signature for one key into a valid signature for another key without disclosing the secret signature keys. We extend and generalize this notion as follows. Intuitively, proxy functions allow one user to correctly decrypt ciphertexts or generate valid signatures on behalf of another user without holding any information about the secret keys of the latter user.

We consider that the proxy functions can be divided into two categories: *bidirectional* and *unidirectional* . The *unidirectional* proxy functions allow one user (U_1) to decrypt ciphertexts or generate signatures corresponding to the secret key of another user (U_2) even if the first user does not hold that secret key. However, the owner of the secret key (U_2) needs a completely different *unidirectional* function if he desires to decrypt ciphertexts or generate signatures on behalf of the first user (U_1). Unlike the *unidirectional* proxy functions, the *bidirectional* ones can be used by both users to decrypt ciphertexts or generate signatures, by transforming the ciphertext/signature for one key into ciphertext/signature for another key. In other words, both users U_1 and U_2 can use the same *bidirectional* proxy function to transform ciphertexts from one key to another key.

The original paper [4] informally defines the notion of *bidirectional* proxy functions and describes two examples of proxy functions: one for encryption, based on El Gamal encryption, and one for signatures. However, both examples are proved to have low security guarantees. Our paper formally defines both the *bidirectional* and *unidirectional* proxy functions for encryption and signature, and their security guarantees (e.g. indistinguishability/unforgeability under various attacks). In addition, this paper presents generic schemes for *bidirectional* and *unidirectional* proxy functions for both public-key and private-key encryption and signature schemes. All generic schemes can be used to transform any standard cryptographic primitive into a proxy function, with a factor of two slowdown. This slowdown is eliminated by the proxy functions specifi-

---

[1]We will also mention yet another off-line variant implicitly studied by [9, 10].

cally designed for a few cryptographic primitives (e.g. El Gamal [12], RSA [25], RSA Hash-and-Sign [2, 1]).

The notion of proxy cryptography can be very useful in cases when one user needs to perform sensitive operations (e.g. ciphertext decryption, signature generation) without holding the necessary secret keys. For example, the president of a company can delegate his signature rights by giving a proxy key to his assistant. The proxy key transforms a signature created by the vice-president into the president's signature, thus allowing the assistent to co-sign only if the document was first signed by the vice-president. Another example is that of a key escrow system [13, 23, 18, 17, 14, 27], where a trusted party can mediate the conflicts between users and the law enforcement agencies. The problem is to allow the law enforcement agency to read messages encrypted for a set of users, for a limited period of time, without knowing the users' secrets. The solution is to locate a key escrow agent between the users and the law enforcement agency, such that it controls which messages are read by the law enforcement agencies. In classic schemes, the users have to give their secret keys to the key escrow agent. Whenever the law enforcement agency wants to reads a message belonging to a user, the key escrow agent decrypts the message and re-encrypts it with the key of the law enforcement agency. In order to prevent the key escrow agent from knowing the secret keys and cleartext messages, we propose that the key escrow agent holds proxy keys that uses proxy functions to transform ciphertext corresponding to user keys into ciphertext corresponding to the law enforcement agency.

The rest of the paper is structured as follows. The next chapter presents other projects that studied the notion of proxy functions. Chapter 3 uses the key escrow scenario to describe the computational model used to define the proxy functions. The next four chapters present the actual *unidirectional* and *bidirectional* functions. The paper ends with some final thoughts about learned lessons and ideas for the future.

2 Related Work

The idea of delegating decryption/signature rights was previously researched and presented in several papers [16, 15, 3, 22, 21, 4]. The goal of the [16, 15] paper is similar to ours. In the context of mobile computing, agents should be able to carry signature functions such that untrusted entities sign on behalf of a user without knowing his key. However, the result of signing a message m is a brand new signature that combines the identities of the original user and the actual signer. Our schemes differ from theirs in that that the new signature is identical to the one that would have been produced by the original delegator. In fact, this indistinguishability is one of the most important feature of our schemes. In [3], the RSA-based unidirectional signature scheme splits the secret key between a client and a server such that neither is able to create a valid key without working together. The security proofs rely on the fact that the server is always trusted, thus obtaining lower levels of security then the ones we propose here. MacKenzie and Reiter [22, 21, 20] consider a similar question of two-party signature generation to the one we consider here. However, their solutions, especially [22] are highly complex and interactive as compared to the notion of unidirectional proxy signatures we propose here (they also have a slightly more sophisticated scenario, where the user has a personal password in addition to the split secret key).

As mentioned, the most closely related work is that of Blaze and Strauss [4] who introduce the notions of bidirectional decryption and bidirectional signature. However, lack of proper definitions makes them consider only the question of changing the *existing* encryption or signature schemes (like ElGamal encryption or Fiat-Shamir signature [11]) into a corresponding proxy primitive, instead of looking at the abstract problem *itself*. As the result, they provide very limited schemes satisfying very weak (semi-formally stated) security properties. We contribute to this work by clarifying and precisely defining the problems at hand (i.e., presenting formal definitions for all *bidirectional* and *unidirectional* proxy functions and their security guarantees), and describe generic as well as specific schemes for both encryption and signature proxy functions.

We briefly consider extensions to the multi-user setting, and use recent results from identity based cryptography [26, 5] to improve the efficiency in this setting. In addition, we adapt the key-insulated model presented in [9, 10] to create offline bidirectional schemes that do not require the proxy agent P to continuously assist the law enforcement agency F.

The unidirectional and bidirectional primitives can be considered as special cases of general threshold cryptography [6, 8]. However, most threshold systems assume a honest majority and work only for $n \geq 3$. Thus, many threshold techniques cannot be applied to a two-party setting. Recently, people have considered two-party primitives in a multi-round setting: GQ, Schnorr [24] and DSA signatures [22], while [19] talks about encryption.

3 Model

For a better understanding and consistency throughout the rest of the paper, we will explain and use the key escrow scenario as a model for our definitions.

The key escrow scenario has four classes of actors: (i) the general users U who delegate their decryption rights, (ii) the law enforcement agency F that tries to decrypt ciphertexts belonging to the general users, (iii) the proxy agent P responsible for helping the latter user to decrypt ciphertexts, and (iv) the legal court that is trusted by everyone. All users register with the key escrow system by providing some kind of secret information to the proxy P. After registration, they are free to send encrypted messages to each other. Whenever the law enforcement agency wants to eavesdrop on the communication between two users, it asks the legal court for a warrant. The legal court creates a time-bounded warrant and gives it to the proxy agent. Then, the proxy agent helps the law enforcement agency to decrypt the ciphertexts belonging to the specified users and period of time. In our model, we will disregard the last actor, because the legal court is not directly involved in the cryptographic part of the protocol.

The next paragraphs informally define the *bidirectional* and *unidirectional* proxy functions for encryption and signature generation and explain how they can be easily used to construct key escrow systems.

Unidirectional encryption proxy function. A unidirectional encryption proxy function is defined as a tuple $\mathcal{E} =$ (UniGen, UniEnc, UniDec, PDec, FDec). The key generation algorithm UniGen generates keys for every general user U. Then, for each user U, it generates two more keys for the proxy P and the user F. The general users encrypt cleartext messages using the UniEnc algorithm and decrypt them using the UniDec algorithm. Whenever the user F wants to decrypt a ciphertext e, it asks the proxy P for help. The proxy P uses PDec to transform the ciphertext e into a different ciphertext e' and sends it to the user F. The user F applies the FDec function to the received ciphertext e' and gets the original cleartext m.

Unidirectional signature proxy function. An unidirectional signature proxy function is defined as a tuple $\mathcal{S} =$ (UniGen, UniSig, UniVer, PSig, FSig). As in the unidirectional encryption case, the key generation algorithm generates keys for every general user U. Then, for each user U, it generates two more keys for the proxy P and the user F. The general users sign messages using the UniSig algorithm and verify them using the UniVer algorithm. When-

ever the user F wants to sign a message m on behalf of a certain user U, it asks the proxy P for help. First, the user F uses FSig to generate a partial signature of the message m. The proxy P transforms the partial signature into a valid signature by applying the PSig on the partial signature.

Bidirectional encryption proxy function. A bidirectional encryption function is defined as a tuple $\mathcal{E} =$ (BiGen, BiEnc, BiDec, Π). The key generation algorithm BiGen creates keys for all users in the system, including the user F. For each pair of keys (k_U, k_F), the BiGen algorithm generates a bidirectional key π. The general users U encrypt messages using BiEnc and decrypt them using BiDec. In order to decrypt a ciphertext belonging to a general user U, the user F asks the proxy P for help. The proxy P uses the bidirectional function Π and the bidirectional key π to transform the ciphertext for user U into ciphertext for user F. After that, the user F can decrypt the new ciphertext with its own key and obtain the cleartext message.

Bidirectional signature proxy function. A bidirectional signature function is defined as a tuple $\mathcal{S} =$ (BiGen, BiSig, BiVer, Π). As in the bidirectional encryption case, the key generation algorithm BiGen creates keys for all users in the system, including the user F. For each pair of keys (k_U, k_F), the BiGen algorithm generates a bidirectional key π. The general users U sign messages using BiSig and verifies the signatures using BiVer. Whenever the user F wants to generate a valid signature for a message m on behalf of a user U, it first generates a signature with its own key and then asks the proxy P for help. The proxy P uses the bidirectional function Π and the bidirectional key π to transform the signature generated by the user F into a signature generated with the user's key.

Table 1 reflects the way the unidirectional and the bidirectional techniques work for both encryption and signatures.

Even though both bidirectional and unidirectional proxy functions achieve the same goal, there are a few notable differences between them. First, the unidirectional proxy functions can be used only one way, from one user to another user. The reverse sense requires a different unidirectional function. The bidirectional proxy functions can be used in both directions. Second, the bidirectional schemes assume that the law enforcement agency (user F) has its own key. The unidirectional schemes do not make this assumption but pay an increased storage requirement price because the user F needs to store one key for each user in the system. In both cases, the proxy P has the increased space problem because it needs to save one key for every

	Bidirectional	*Unidirectional*
Encryption	$U(sk_U) \quad\quad P(\pi) \quad\quad F(sk_F)$ $e=Enc_U(m) \longrightarrow e'=\pi(e) \longrightarrow e'=Enc_F(m)$	$U(DK_U)$ $P(DK_P) \quad\quad F(DK_F)$ $e=Enc_U(m) \longrightarrow PDec(e) \longrightarrow FDec(PDec(e))=m$
Signature	$F(sk_F) \quad\quad P(\pi) \quad\quad U(sk_U)$ $s'=Sig_F(m) \longrightarrow s'=\pi(s) \longrightarrow s=Sig_U(m)$	$U(sk)$ $F(sk_F) \quad\quad P(sk_P)$ $m \longrightarrow FSig(m) \longrightarrow s=Psig(FSig(m))$

Table 1. Unidirectional vs. Bidirectional techniques

user. This problem can be very important in systems where the number of general users is extremely large. A solution is offered by the identity-based primitives [5], where the proxy needs to save only a share of the master secret key. Third, in both cases, revocation is easily achieved by having the third party P refuse to help.

The unidirectional and bidirectional schemes described in this paper require the proxy agent P to continuously assist the law enforcement agency F when decrypting ciphertext or generating valid signatures on behalf of a user. The key-insulated model presented in [9, 10] can be easily extended such that the proxy agent P helps the law enforcement agency only once, at the beginning of its warrant. The key-insulated model has two actors, the proxy agent P and the user U. The user U updates its secret key using the index of the current time period and some information provided by a third party (P). Our model adds another player, the law enforcement agency F, that receives from the proxy P the user's key for an unused period of time T_0. Similar to the original key-insulated model, the proxy P helps the user to update its key. In addition, the proxy P helps the law enforcement agency F to compute the user's key for time period T_i if presented with an warrant for the time T_i. All primitives presented in [9, 10] can be easily extended for our offline model.

With this model in mind, we introduce in the next sections the *unidirectional* and *bidirectional* encryption and signature schemes. For each scheme, we give formal definitions, present one generic scheme and several specific schemes, and prove their security guarantees. For simplicity, all general definitions will be given in the context of public-key cryptography.

4 Unidirectional Encryption Primitives

Definition 1 *A* unidirectional *encryption scheme consists of five algorithms,* $\mathcal{E} =$ (UniGen, UniEnc, UniDec, PDec, FDec).

The generation algorithm UniGen(1^k) outputs a tuple of keys (EK,DK) for each general user U. EK is the encryption key and DK is the decryption key. For each secret key DK, the key generation algorithm creates two secret keys DK_P and DK_F for the proxy P, and respectively the user F. For simplicity, the definitions given in table 2 show that the key generation algorithm outputs only user keys, even though it also builts the keys for the proxy P and the user F.

UniEnc$_{EK}$ is the encryption algorithm and encrypts a message m from the corresponding message space M (e.g., $\{0,1\}^k$) as $e = $ UniEnc$_{EK}(m)$. The decryption algorithm UniDec$_{DK}$ is a deterministic decryption algorithm that takes the ciphertext e, the secret key DK, and outputs $m \in M$ (or invalid in case e was an improper ciphertext). The *correctness* property of encryption states that UniDec(UniEnc(m)) $= m$, for any message m and pair of keys (EK, DK). The function PDec uses the secret key of the proxy P, DK_P, to transform a ciphertext e into ciphertext e'. The FDec function takes this ciphertext e' and the secret key DK_F of the user F and produces the original message $m \in M$ or invalid if the ciphertext e' was not correct. The *correctness* property specifies that FDec(PDec(UniEnc(m))) $= m$, for any m and (EK, DK).

Informally, a unidirectional encryption scheme is considered to be secure if none of the participating entities (user F, proxy P, user U) can break it even if they hold extra secrets. For simplicity, the definitions presented in table 2 will be specific to the CCA2 security for public key encryption [2]. In our definitions, the proxy agent P gets only

---

[2]CPA and ONE-WAY security definitions are given when necessary.

Definition 2 *Let $\mathcal{E} = $ (UniGen, UniEnc, UniDec, PDec, FDec) be an unidirectional encryption scheme.*

1. *$\mathcal{E}$ is* CCA2 *secure against the proxy* P *if* $|\,\mathsf{Succ}_{\mathsf{P},\mathcal{E}}(1^k) - 1/2\,|$ *is negligible,* $\mathsf{Succ}_{\mathsf{P},\mathcal{E}}$ *is defined as below,* PDec *is a deterministic algorithm, and the proxy* P *never submits* $\mathsf{PDec}(\mathsf{UniEnc}_{\mathsf{EK}}(m_b))$ *to the* FDec *oracle.*

$$\mathsf{Succ}_{\mathsf{P},\mathcal{E}} \;\stackrel{\text{def}}{=}\; \Pr\left[\,b = \tilde{b}\;\middle|\;\begin{array}{c}(\mathsf{EK},\mathsf{DK}) \leftarrow \mathsf{UniGen}(1^k), (m_0,m_1) \leftarrow \mathsf{P}^{\mathsf{FDec}}(\mathsf{EK},\mathsf{DK}_{\mathsf{P}}),\\ b \leftarrow \{0,1\}, \tilde{b} \leftarrow \mathsf{P}^{\mathsf{FDec}}(\mathsf{EK},\mathsf{DK}_{\mathsf{P}},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\,\right]$$

2. *$\mathcal{E}$ is* CCA2 *secure against the user* F *if* $|\,\mathsf{Succ}_{\mathsf{F},\mathcal{E}}(1^k) - 1/2\,|$ *is negligible,* $\mathsf{Succ}_{\mathsf{F},\mathcal{E}}$ *is defined as below, and the user* F *cannot submit the challenge* $\mathsf{UniEnc}_{\mathsf{EK}}(m_b)$ *to the* PDec *oracle.*

$$\mathsf{Succ}_{\mathsf{F},\mathcal{E}} \;\stackrel{\text{def}}{=}\; \Pr\left[\,b = \tilde{b}\;\middle|\;\begin{array}{c}(\mathsf{EK},\mathsf{DK}) \leftarrow \mathsf{UniGen}(1^k), (m_0,m_1) \leftarrow \mathsf{F}^{\mathsf{PDec}}(\mathsf{EK},\mathsf{DK}_{\mathsf{F}}),\\ b \leftarrow \{0,1\}, \tilde{b} \leftarrow \mathsf{F}^{\mathsf{PDec}}(\mathsf{EK},\mathsf{DK}_{\mathsf{F}},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\,\right]$$

3. *$\mathcal{E}$ is* CCA2 *secure against any user* U *if* $|\,\mathsf{Succ}_{\mathsf{U},\mathcal{E}}(1^k) - 1/2\,|$ *is negligible,* $\mathsf{Succ}_{\mathsf{U},\mathcal{E}}$ *is defined as below, and the user* U *cannot submit the challenge* $\mathsf{UniEnc}_{\mathsf{EK}}(m_b)$ *to the decryption oracle* UniDec.

$$\mathsf{Succ}_{\mathsf{U},\mathcal{E}} \;\stackrel{\text{def}}{=}\; \Pr\left[\,b = \tilde{b}\;\middle|\;\begin{array}{c}(\mathsf{EK},\mathsf{DK}) \leftarrow \mathsf{UniGen}(1^k), (m_0,m_1) \leftarrow \mathsf{U}^{\mathsf{UniDec}}(\mathsf{EK}),\\ b \leftarrow \{0,1\}, \tilde{b} \leftarrow \mathsf{U}^{\mathsf{UniDec}}(\mathsf{EK},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\,\right]$$

Table 2. Online encryption definitions.

oracle access to the FDec. In fact, P does not need access to UniDec because it can simulate it by itself. The only conditions necessary are that PDec is a deterministic algorithm, and the proxy P never submits $\mathsf{PDec}(\mathsf{UniEnc}_{\mathsf{EK}}(m_b))$ to the FDec oracle.

For each of the unidirectional schemes described next, we will prove that they are secure according to all three definitions.

4.1 Unidirectional Generic Encryption Scheme

We first present a unidirectional generic technique that transforms a general encryption scheme $\mathcal{E} = $ (Enc-Gen, Enc, Dec) into an unidirectional encryption scheme $\mathcal{E}' = $ (UniGen, UniEnc, UniDec, PDec, FDec). The key generation algorithm UniGen generates two pairs of keys $(\mathsf{EK}_1, \mathsf{DK}_1, \mathsf{EK}_2, \mathsf{DK}_2)$ by running the Enc-Gen algorithm twice. The user U keeps both keys, while proxy P and the user F get $(\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{DK}_1)$, respectively $(\mathsf{EK}_1, \mathsf{EK}_2, \mathsf{DK}_2)$. We define $\mathsf{DK}_{\mathsf{P}} = \mathsf{DK}_1$ and $\mathsf{DK}_{\mathsf{F}} = \mathsf{DK}_2$. The encryption algorithm UniEnc is equivalent to encrypting the message with the two keys EK_1 and EK_2: $\mathsf{UniEnc}(m) = \mathsf{Enc}_1(\mathsf{Enc}_2(m)) = e$. The unidirectional decryption algorithm UniDec decrypts the ciphertext e by applying the original decryption algorithm Dec twice: $m = \mathsf{Dec}_2(\mathsf{Dec}_1(e))$. The proxy P uses the function PDec

to transform the ciphertext e into ciphertext e' by decrypting once with its key $\mathsf{DK}_{\mathsf{P}} = \mathsf{DK}_1$: $e' = \mathsf{Dec}_1(e)$. The user F uses FDec to transform the ciphertext e' into the initial message m (or invalid) by decrypting once with its key $\mathsf{DK}_{\mathsf{F}} = \mathsf{DK}_2$: $m = \mathsf{Dec}_2(e')$.

The double encryption can be also defined using two different encryption schemes $\mathcal{E}_1 = (\mathsf{Enc\text{-}Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$, $\mathcal{E}_2 = (\mathsf{Enc\text{-}Gen}_2 \mathsf{Enc}_2, \mathsf{Dec}_2)$. In this case, the unidirectional encryption scheme $\mathcal{E}' = $ (UniGen, UniEnc, UniDec, PDec, FDec) is defined as:

- $\mathsf{UniGen}(1^k) = (\mathsf{Enc\text{-}Gen}_{\mathcal{E}_1}(1^k), \mathsf{Enc\text{-}Gen}_{\mathcal{E}_2}(1^k))$

- $\mathsf{UniEnc}(m) = \mathsf{Enc}_{\mathcal{E}_1}(\mathsf{Enc}_{\mathcal{E}_2}(m))$

- $\mathsf{UniDec}(e) = \mathsf{Dec}_{\mathcal{E}_2}(\mathsf{Dec}_{\mathcal{E}_1}(e))$

- $\mathsf{PDec}(e) = \mathsf{Dec}_{\mathcal{E}_1}(e)$

- $\mathsf{FDec}(e') = \mathsf{Dec}_{\mathcal{E}_2}(e')$

For simplicity, we assume that both encryption schemes are identical. Next, we prove that the generic unidirectional encryption scheme is secure according to our definitions. We make the assumption that the initial encryption scheme we started from is CCA2 and show that the new unidirectional encryption scheme is also CCA2 . The proofs are in the Appendix A.1.

Theorem 1 *Let's consider a standard encryption scheme* $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. *Based on $\mathcal{E}$, we build an unidirectional encryption scheme* $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$. *If $\mathcal{E}$ is* CCA2 *secure, than $\mathcal{E}'$ is also* CCA2 *secure against (1) the proxy* P, *(2) the user* F, *and (3) any user* U.

The generic scheme is twice slower than the original scheme it started from. In order to eliminate this slowdown, we developed a few specific unidirectional encryption functions based on El Gamal, RSA, and IBE.

4.2 Unidirectional El Gamal Encryption Scheme

Let's assume that we have an El Gamal encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm outputs the public key $\mathsf{EK} = (g, p, q, y)$ and the secret key $\mathsf{DK} = (x)$, where p is a prime number, g is a generator for the $\mathbb{Z}_p^*$, x is randomly chosen from $\mathbb{Z}_q$, and $y = g^x \bmod p$. The encryption algorithm is defined as $e = \mathsf{Enc}_{\mathsf{EK}}(m) = (g^r \bmod p, mg^{xr} \bmod p)$, where r is chosen at random from $\mathbb{Z}_q$. The decryption algorithm computes the message m from e by dividing mg^{xr} to $(g^r)^x \bmod p$.

The unidirectional El Gamal encryption scheme is defined as $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$. For each user U, the key generation algorithm $\mathsf{UniGen}(1^k)$ generates a public-key pair $(\mathsf{EK}, \mathsf{DK})$ and splits the secret key $\mathsf{DK} = x$ into two parts x_1 and $= x_2$ such that $x = x_1 + x_2$. The proxy P receives $\mathsf{DK}_\mathsf{P} = x_1$ and the user F receives $\mathsf{DK}_\mathsf{F} = x_2$. The encryption UniEnc and the decryption UniDec algorithms are identical to the standard algorithms: Enc and Dec. The transformation algorithms PDec and FDec are equivalent to Dec under x_1, respectively x_2. The unidirectional encryption scheme is correct because $\mathsf{FDec}_{x_2}(\mathsf{PDec}_{x_1}(\mathsf{Enc}_y(m))) = \mathsf{FDec}_{x_2}(mg^{xr}/(g^r)^{x_1}) = mg^{x_2r}/(g^r)^{x_2} = m$.

According to our definitions, the next theorem proves that the unidirectional El Gamal is as secure as the original El Gamal scheme. The proofs are presented in Appendix A.2.

Theorem 2 *Let $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$ be an unidirectional El Gamal encryption scheme. $\mathcal{E}'$ is* CPA *secure against (1) the proxy* P, *(2) the user* F, *and (3) any user* U.

4.3 Unidirectional RSA Encryption Scheme

Let's assume we have the RSA encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. The RSA key generation algorithm outputs the public key $\mathsf{EK} = (e, N)$ and the secret key $\mathsf{SK} = (d, N, \varphi(N))$, where $ed = 1 \bmod \varphi(N)$, $N = pq$, p, q are two large primes and φ is the Euler totient function. The encryption is defined as $\mathsf{Enc}_{\mathsf{EK}}(m) = m^e \bmod N = c$. The decryption algorithm is $\mathsf{Dec}_{\mathsf{EK}}(c) = c^d \bmod N = m$.

The unidirectional key generation algorithm $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$. For each user U, the key generation algorithm $\mathsf{UniGen}(1^k)$ generates a public-key pair $(\mathsf{EK}, \mathsf{DK})$ and splits the secret key into two parts d_1 and d_2 such that $d = d_1 d_2 \bmod \varphi(N)$. The proxy P gets $\mathsf{DK}_\mathsf{P} = d_1$ and the user F gets $\mathsf{DK}_\mathsf{F} = d_2$. The encryption UniEnc and the UniDec algorithms are identical to the original algorithms Enc and Dec. The transformation functions PDec and FDec execute the Dec decryption algorithm with keys d_1 and respectively d_2. The correctness of unidirectional RSA is given by the equality $\mathsf{FDec}_{d_2}(\mathsf{PDec}_{d_1}(\mathsf{Enc}_e(m))) = m$.

The original RSA scheme is OW-CPA secure. Thus, we will prove in the next theorem that unidirectional RSA is also OW-CPA secure. The proofs are in Appendix A.3.

Theorem 3 *Let $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$ be an unidirectional RSA encryption scheme. $\mathcal{E}'$ is* ONE-WAY *secure against (1) the proxy* P, *(2) the user* F, *and (3) any user* U.

4.4 Unidirectional Identity-Based Encryption Scheme

Our specific IBE scheme is a slightly modification of the original IBE scheme introduced by [5]. The original IBE scheme uses a bilinear map $\hat{e}$ defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups of order q and q is a large prime number. This means that $\hat{e}(aP, bQ) = \hat{e}(aQ, bP) = \hat{e}(P, Q)^{ab}$, where $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$. The original scheme bases its security on the computational Bilinear Diffie-Hellman problem. In order to obtain a homomorphic scheme, we make a stronger assumption (decisional Bilinear Diffie-Hellman problem), eliminate the use of a hash function, require that the messages are $m \in \mathbb{G}_2$, and replace the XOR operation by multiplication.

Our scheme is defined as the tuple $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm creates the master secret key s and the master public key $P_{pub} = sP$. For every user U, the Extract

algorithm takes as input the user's ID and returns a secret key DK $= sID$ and a public key equal to the ID. The user's ID is actually defined to be the hash value of the "real" ID. The encryption algorithm Enc takes the message m and the public key ID as the input and creates the ciphertext $\langle U, V \rangle = \langle rP, m\hat{e}(rID, sP)) \rangle$. The decryption algorithm Dec computes $V/\hat{e}(sID, U)) = m$. The IBE scheme is CPA secure if the Bilinear Diffie-Hellman (BDH) problem is hard.

Decisional Bilinear Diffie-Hellam Problem (dBDH). Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map and let P be a generator for $\mathbb{G}_1$. The decisional BDH problem $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is defined as follows: Given $\langle P, aP, bP, cP \rangle$ for some $a, b, c \in \mathbb{Z}_q^*$, it is hard to differentiate $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$ from a random $g \in \mathbb{G}_2$.

Based on our specific IBE scheme $\mathcal{E}$, we will build an unidirectional IBE scheme $\mathcal{E}' = $ (UniGen, UniEnc, UniDec, PDec, FDec). The key generation algorithm UniGen uses the original key generation algorithm Enc-Gen to create the master secret key s and the master public key $P_{pub} = sP$. The master secret key is split in two parts s_1 and s_2, and each part is given to the proxy P and the user F. The encryption and decryption algorithms are identical with the original ones. PDec is defined as PDec$(U, V) = \langle U, V/\hat{e}(rP, s_1 ID) \rangle = (U, V')$. The user F uses the ciphertext generated by PDec and the function FDec to compute the cleartext message m by computing $V'/\hat{e}(rP, s_2 ID)$.

Theorem 4 *Let $\mathcal{E}' = $ (UniGen, UniEnc, UniDec, PDec, FDec) be an unidirectional IBE encryption scheme. $\mathcal{E}'$ is CPA secure against (1) the proxy P, (2) the user F, and (3) any user U.*

In all the other unidirectional schemes, the proxy P has the increased space problem because it needs to save one key for every user. Our modified homomorphic IBE scheme solves this problem by allowing the proxy agent P to save only a single share of the master key for the entire system.

5   Unidirectional Signature Primitives

Definition 3 *A* unidirectional *signature scheme consists of five algorithms:* $\mathcal{S} = $ (UniGen, UniSig, UniVer, PSig, FSig).

The generation algorithm UniGen(1^k) outputs a tuple

of keys (SK, VK) for each user U. VK is the verification key and SK is the signing key. SK is used to generate the keys SK_P and SK_F given to the proxy P, respectively the user F. The signature algorithm UniSig signs a message $m \in M$ (e.g. $\{1, 0\}^k$), $s = $ UniSig$_{SK}(m)$ using the secret key SK. The signature is formed by the tuple (m, s). The verification algorithm UniVer uses the public key to verify that a signature (m, s) is valid. The verification algorithm output *succeed* if the signature is correct and *fail* otherwise. The *correctness* property requires that UniVer(UniSig(m)) $= succeed$. The proxy P uses the function PSig to generate a partial signature of a message $m \in M$ based on SK_P. The user F uses FSig to generate a partial signature of a message $m \in M$ based on SK_F. PSig(FSig(m)) form a complete signature of the message m.

We define the unidirectional signature scheme to be safe if neither entity (proxy P, user F, user U) can generate alone valid signatures under key SK, even if they know SK_P or SK_F and any of the available public information. The formal definitions are given in table 3.

5.1   Unidirectional Generic Signature Scheme

This generic scheme transforms any given signature primitive into a unidirectional generic signature scheme. Let's assume that $\mathcal{S} = $ (Sig-Gen, Sig, Ver) is a standard signature scheme. The new unidirectional generic signature scheme is $\mathcal{S}' = $ (UniGen, UniSig, UniVer, PSig, FSig). The generation algorithm UniGen generates two pairs of keys (SK_1, VK_1, SK_2, VK_2) for each user U. The secret key for the unidirectional signature scheme is SK $= (SK_1, SK_2)$. The proxy P gets $(VK_1, VK_2, SK_P = SK_1)$ and the user F gets $(VK_1, VK_2, SK_F = SK_2)$. The signature algorithm UniSig generates a valid signature for a message $m \in M$ by applying the signature Sig twice: $s = (s_1, s_2) = $ Sig$_1(m)$Sig$_2(m)$. Similarly, the verification algorithm UniVer verifies whether the signatures generated by UniSig are valid by applying the original verification algorithm Ver algorithm twice: Ver$_1(s_1)$Ver$_2(s_2)$. The proxy P uses PSig to generate part of the total *unidirectional* signature: Sig$_1(m)$, while the user F uses FSig $= $ Sig$_2(m)$ to generate the entire signature together with PSig.

According to our definitions, the generic unidirectional signature scheme is secure if the following theorem is proved to be true. The actual proofs are contained in Appendix B.1.

Definition 4 *Let* $\mathcal{S} = $ (UniGen, UniSig, UniVer, PSig, FSig) *be an unidirectional signature scheme.*

1. *$\mathcal{S}$ is* UF-CMA ***against the proxy*** P *if* $|\mathsf{Succ}_{\mathsf{P},\mathcal{S}}(1^k)|$ *is negligible,* $\mathsf{Succ}_{\mathsf{P},\mathcal{S}}$ *is defined as below, and the proxy* P *is not allowed to ask the* FSig *oracle for* UniSig(m).

$$\mathsf{Succ}_{\mathsf{P},\mathcal{S}} \;\stackrel{\text{def}}{=}\; \Pr\left[\, \mathsf{UniVer}(m,s) = succeed \,\middle|\, (\mathsf{SK},\mathsf{VK}) \leftarrow \mathsf{UniGen}(1^k), (m,s) \leftarrow \mathsf{P}^{\mathsf{FSig}}(\mathsf{SK}_\mathsf{P},\mathsf{VK}) \,\right]$$

2. *$\mathcal{S}$ is* UF-CMA ***against the user*** F *if* $|\mathsf{Succ}_{\mathsf{F},\mathcal{S}}(1^k)|$ *is negligible,* $\mathsf{Succ}_{\mathsf{F},\mathcal{S}}$ *is defined as below, and the user* F *is not allowed to ask the signature oracle for* UniSig(m).

$$\mathsf{Succ}_{\mathsf{F},\mathcal{S}} \;\stackrel{\text{def}}{=}\; \Pr\left[\, \mathsf{UniVer}(m,s) = succeed \,\middle|\, (\mathsf{SK},\mathsf{VK}) \leftarrow \mathsf{UniGen}(1^k), (m,s) \leftarrow \mathsf{F}^{\mathsf{UniSig}}(\mathsf{SK}_\mathsf{F},\mathsf{VK}) \,\right]$$

3. *$\mathcal{S}$ is* UF-CMA ***against any user*** U *if* $|\mathsf{Succ}_{\mathsf{U},\mathcal{S}}(1^k)|$ *is negligible for any PPT adversary* U, $\mathsf{Succ}_{\mathsf{U},\mathcal{S}}$ *is defined as below, and the user* U *is not allowed to ask the signature oracle for* UniSig(m).

$$\mathsf{Succ}_{\mathsf{U},\mathcal{S}} \;\stackrel{\text{def}}{=}\; \Pr\left[\, \mathsf{UniVer}(m,s) = succeed \,\middle|\, (\mathsf{SK},\mathsf{VK}) \leftarrow \mathsf{UniGen}(1^k), (m,s) \leftarrow \mathsf{U}^{\mathsf{UniSig}}(\mathsf{VK}) \,\right]$$

Table 3. Online signature definitions.

Theorem 5 *Let* $\mathcal{S} = $ (Sig-Gen, Sig, Ver) *be a standard signature scheme. Let's consider* $\mathcal{S}' = $ (UniGen, UniSig, UniVer, PSig, FSig) *a unidirectional signature scheme constructed as described above, based on* $\mathcal{S}$. *If* $\mathcal{S}$ *is* UF-CMA *, than* $\mathcal{S}'$ *is* UF-CMA *against (1) the proxy* P, *(2) the user* F, *and (3) any user* U.

The unidirectional generic signature scheme has two main performance disadvantages. First, the size of the secret key increases. Each user no longer has one, but two keys. Second, the number of operations performed when signing and verifying doubles. In order to improve these numbers, we developed an efficient unidirectional signature scheme based on RSA-Hash.

5.2 Unidirectional RSA-Hash Signature Scheme

Let's assume that we have $\mathcal{S} = $ (Sig-Gen, Sig, Ver) a standard RSA-Hash(Full Domain Hash) [1] signature. Sig-Gen generates the public key $\mathsf{VK} = (e, N)$ and the secret key $\mathsf{SK} = (d, \varphi(N))$. The signature function is defined as $\mathsf{Sig} = hash(m)^d \bmod N = s$, where $hash$ is a hash function associated with Sig. The verification returns succeed if $s^e = hash(m) \bmod N$. Otherwise, it returns fail.

The standard RSA-Hash signature scheme $\mathcal{S}$ is transformed into a unidirectional signature scheme $\mathcal{S}' = $ (UniGen, UniSig, UniVer, PSig, FSig) by the following steps. The key generation algorithm UniGen generates keys

for all users U by executing Sig-Gen and then splits each secret key d in two parts $d = d_1 + d_2 \bmod \varphi(n)$. d_1 becomes the key SK_P of the proxy P and d_2 becomes the key SK_F of the user F. The signature and verification algorithms UniSig and UniVer are identical with the original algorithms Sig and Ver. The user F uses the FSig function to generate one part of the unidirectional RSA-Hash signature by computing $s' = \mathsf{Sig}_{d_2}(m)$. The proxy P uses the PSig function to generate the other part of the unidirectional RSA signature by computing $s = \mathsf{Sig}_{d_1}(m)$. The signature is formed by s and s'.

The standard RSA-Hash signature scheme is existentially unforgeable against chosen message attacks. Thus, we formally prove in the next theorem that the unidirectional RSA-Hash scheme holds the same level of security. The actual proofs are in Appendix B.2.

Theorem 6 *Let* $\mathcal{S} = $ (Sig-Gen, Sig, Ver,) *be a classic RSA-Hash signature scheme. Let's consider that* $\mathcal{S}' = $ (UniGen, UniSig, UniVer, PSig, FSig) *is an unidirectional RSA-Hash signature scheme constructed as above.* $\mathcal{S}'$ *is* UF-CMA *against (1) the proxy* P, *(2) the user* F, *and (3) all users* U.

The probabilistic RSA-Hash described by [7] has better security that the RSA-Hash function used above, and can be transformed into an unidirectional primitive if we allow the user F to generate the necessary randomness.

6 Bidirectional Encryption Primitives

Definition 5 *A bidirectional encryption scheme consists of four algorithms:* $\mathcal{E} = (\mathsf{BiGen}, \mathsf{BiEnc}, \mathsf{BiDec}, \Pi)$.

The key generation algorithm BiGen outputs one pair of keys $(\mathsf{EK_U}, \mathsf{DK_U})$ for each user U. In addition, it generates keys for the user F, $(\mathsf{EK_F}, \mathsf{DK_F})$. After that, it creates one bidirectional key π for each user U. The bidirectional keys π are given to proxy P. The encryption algorithm $\mathsf{BiEnc_{EK}}$ takes as input a message m to be encrypted and a public key EK and outputs the ciphertext $e = \mathsf{BiEnc_{EK}}(m)$. $\mathsf{BiDec_{DK}}$ is the deterministic decryption algorithm that takes the ciphertext e, the secret key DK corresponding to the public key, and produces $m \in M$ (or invalid in case e was an improper ciphertext). The *correctness* property of encryption states that $\mathsf{BiDec}(\mathsf{BiEnc}(m)) = m$, for any m and $(\mathsf{EK}, \mathsf{DK})$. Π is the bidirectional function and transforms ciphertexts encrypted with one key $(\mathsf{EK_U})$ into ciphertexts encrypted with another key $(\mathsf{EK_F})$.

We define the bidirectional encryption schemes to be secure if neither the third party (proxy P) nor the users (U, F) can attack the scheme. For simplicity, the definition presented in table 4 uses the CCA2 level of security. For technical reasons, we assume that there exists an efficient algorithm that evaluates the relation $R_\pi(e, e')$ to true or false, where $e = \mathsf{BiEnc}(m)$ is the original chiphertext and $e' = \Pi(e)$ is the modified ciphertext computed by the proxy P. The output of the algorithm is true, it must be the case that $\mathsf{Dec_{EK_U}}(e) = \mathsf{Dec_{EK_F}}(e')$. Having such an algorithm, we allow the proxy P has oracle access to $\mathsf{BiDec_F}$ because it can simulate oracle access to $\mathsf{BiDec_U}$ by itself. In addition, we restrict its access to the $\mathsf{BiDec_F}$ by not letting the proxy P to submit to the oracle a ciphertext e' such that $R_\pi(e, e') = true$.

6.1 Bidirectional Generic Encryption Scheme

In this section, we present a generic implementation of a bidirectional encryption scheme based on standard encryption schemes. Let's assume that we have an encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. We transform $\mathcal{E}$ into a bidirectional encryption scheme $\mathcal{E}' = (\mathsf{BiGen}, \mathsf{BiEnc}, \mathsf{BiDec}, \Pi)$ by following the next steps. For every user U, the generation algorithm BiGen executes the original generation algorithm Enc-Gen to generate three pairs of keys (k_1, k_2, k_3), where each $k_i = (\mathsf{EK}_i, \mathsf{DK}_i)$. The users U, P, and F receive each two pairs of keys such that any two entities have only one pair of keys in common.

For example, the keys of user U are $\mathsf{EK_U} = (\mathsf{EK}_1, \mathsf{EK}_2)$, $\mathsf{DK_U} = (\mathsf{DK}_1, \mathsf{DK}_2)$, the keys of proxy P are $\mathsf{EK_P} = (\mathsf{EK}_2, \mathsf{EK}_3)$, $\mathsf{DK_P} = (\mathsf{DK}_2, \mathsf{DK}_3)$, and the keys of user F are $\mathsf{EK_F} = (\mathsf{EK}_1, \mathsf{EK}_3)$, $\mathsf{DK_F} = (\mathsf{DK}_1, \mathsf{DK}_3)$. In the context of bidirectional encryption, we say that both users U and F have private keys, while the proxy P has the bidirectional key π. The encryption algorithm BiEnc performs double encryption $e = \mathsf{BiEnc_U}(m) = \mathsf{Enc}_1(\mathsf{Enc}_2(m))$. Similarly, the decryption algorithm Π is defined as double decryption $m = \mathsf{BiDec_U}(e) = \mathsf{Dec}_2(\mathsf{Dec}_1(e))$. The proxy function Π transforms the ciphertext encrypted with the user U's key into ciphertext encrypted with the user F's key. The first step is to decrypt the ciphertext $e = \mathsf{BiEnc_U}(m) = \mathsf{Enc}_1(\mathsf{Enc}_2(m))$ by executing $e' = \mathsf{Dec}_1(e)$. Then, it encrypts e' with the other half of the key $e'' = \mathsf{Enc}_3(e')$. The result is $e'' = \mathsf{BiEnc_F}(m)$.

The generic bidirectional encryption scheme described above is secure if no adversary (proxy P, user F, userU) is able to break it. Let's assume that the initial encryption scheme is CCA2 secure. We will show that in this case, the bidirectional encryption is also CCA2 secure. The proofs are in Appendix C.1.

Theorem 7 *Let's consider a standard encryption scheme* $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. *Based on* $\mathcal{E}$, *we build an unidirectional encryption scheme* $\mathcal{E}' = (\mathsf{UniGen}, \mathsf{UniEnc}, \mathsf{UniDec}, \mathsf{PDec}, \mathsf{FDec})$. *If* $\mathcal{E}$ *is* CCA2 *secure, than* $\mathcal{E}'$ *is also* CCA2 *secure against (1) the proxy* P, *(2) the user* F, *and (3) any user* U.

6.2 Bidirectional El Gamal Encryption Scheme

Let's assume we have the El Gamal encryption scheme $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm of the original El Gamal encryption scheme outputs the public key $\mathsf{EK} = (g, p, q, y)$ and the secret key $\mathsf{DK} = x$, and the public key is $y = g^x \bmod p$. The encryption algorithm is defined as $e = \mathsf{Enc_{EK}}(m) = (g^r \bmod p, mg^{xr} \bmod p)$, where r is chosen at random from $\mathbb{Z}_q$. The decryption algorithm computes the message m from e by dividing mg^{xr} to $(g^r)^x \bmod p$.

Based on $\mathcal{E}$, we will build the bidirectional El Gamal encryption scheme $\mathcal{E}' = (\mathsf{BiGen}, \mathsf{BiEnc}, \mathsf{BiDec}, \Pi)$ by following the next steps. The generation algorithm BiGen generates the keys for all users U and F by executing Enc-Gen twice. Let's assume that the generated keys are $(\mathsf{DK_U} = x_1, \mathsf{EK_U} = g^{x_1})$, $(\mathsf{DK_F} = x_2, \mathsf{EK_F} = g^{x_2})$. After this, it computes one proxy key π for every user U: $\pi = x_2 - x_1$. The encryption algorithm

Table 4. Bidirectional encryption definitions.

BiEnc encrypts messages $m \in M$ by executing Enc: $e = \mathsf{BiEnc}(m) = \mathsf{Enc}(m)$. The decryption algorithm BiDec uses Dec to decrypt the ciphertext e: $\mathsf{BiDec}(e) = \mathsf{Dec}(e)$. The proxy function Π transforms messages encrypted with the secret key of the user U (DK_U) into message encrypted with the secret key of the user F (DK_F). We define Π to be $\Pi(\mathsf{BiEnc}_{x_1}(m), \pi) = (g^r, g^{rx}m(g^r)^\pi)$. The function is correct because $\Pi(\mathsf{BiEnc}_{x_1}(m), \pi) = (g^r, g^{rx}mg^{r(x_2 - x_1)}) = (g^r, g^{rx_2}m) = \mathsf{BiEnc}_{x_2}(m)$.

The proxy function described above is secure if neither the proxy P nor the user F can distinguish between encryptions of two messages even if provided with the proxy key. In addition, it should retain the same level of security as the original El Gamal scheme against all other users U. The proofs are described in Appendix C.2.

Theorem 8 *Let $\mathcal{E} = $ (BiGen, BiEnc, BiDec, Π) be an bidirectional El Gamal encryption scheme. $\mathcal{E}$ is CPA secure against (1) the proxy P, (2) the user F, and (3) any user U.*

7 Bidirectional Signature Primitives

Definition 7 *A* bidirectional *signature scheme consists of four algorithms: $\mathcal{S} = $ (BiGen, BiSig, BiVer, Π).*

The key generation algorithm $\mathsf{BiGen}(1^k)$, where k is the security parameter, generates keys for all users, including the user F. For example, the user U gets the keys $(\mathsf{SK}_\mathsf{U}, \mathsf{VK}_\mathsf{U})$ and the user F gets $(\mathsf{SK}_\mathsf{F}, \mathsf{VK}_\mathsf{F})$. The generation algorithm is also computing one bidirectional key π for every user and gives it to proxy P. The signature algorithm BiSig signs a message $m \in M$ (e.g. $\{1,0\}^k$), $s = \mathsf{BiSig}_{\mathsf{SK}}(m)$ using a secret key SK. The signature is formed by the tuple (m, s). The signature (m, s) is verified by the verification algorithm BiVer. The verification algorithm outputs succeed if the signature is correct and fail otherwise. The *correctness* property requires that $\mathsf{BiVer}(\mathsf{BiSig}(m)) = succeed$. The proxy function Π uses the bidirectional key π to transform a signature generated with a secret key into a signature generated with another secret key.

A bidirectional encryption scheme defined as above is considered to be safe if it can not be successfully attacked by any user (U, F) or by the third party (P). The formal definitions are presented in table 5. We assume that partial signature contains the message.

Next, we present a few bidirectional signature schemes that respect the above definition of security.

Definition 8 *Let* $\mathcal{S} = (\mathsf{BiGen}, \mathsf{BiSig}, \mathsf{BiVer}, \Pi)$ *be an bidirectional signature scheme.*

1. *$\mathcal{S}$ is* UF ***against the proxy*** P *if* $|\mathsf{Succ}_{\mathsf{P},\mathcal{S}}(1^k)|$ *is negligible, where* $\mathsf{Succ}_{\mathsf{P},\mathcal{S}}|$ *is defined as below and* P *is not allowed to ask the signature oracle for* $\mathsf{BiSig}(m)$.

$$\mathsf{Succ}_{\mathsf{P},\mathcal{S}} \ \stackrel{\text{def}}{=} \ \Pr\left[\ \mathsf{BiVer}(m,s) = succeed \ \middle|\ \begin{array}{l} (\mathsf{SK}_\mathsf{U}, \mathsf{VK}_\mathsf{U}, \mathsf{SK}_\mathsf{F}, \mathsf{VK}_\mathsf{F}, \pi) \leftarrow \mathsf{BiGen}(1^k) \\ (m,s) \leftarrow \mathsf{P}^{\mathsf{BiSig}_\mathsf{F}}(\mathsf{VK}_\mathsf{U}, \mathsf{VK}_\mathsf{F}, \pi) \end{array}\right]$$

2. *$\mathcal{S}$ is* UF ***against the user*** F *if* $|\mathsf{Succ}_{\mathsf{F},\mathcal{S}}(1^k)|$ *is negligible,* $\mathsf{Succ}_{\mathsf{F},\mathcal{S}}|$ *is defined as below and* F *is not allowed to ask* P *for* $\Pi(m)$.

$$\mathsf{Succ}_{\mathsf{F},\mathcal{S}} \ \stackrel{\text{def}}{=} \ \Pr\left[\ \mathsf{BiVer}(m,s) = succeed \ \middle|\ \begin{array}{l} (\mathsf{SK}_\mathsf{U}, \mathsf{VK}_\mathsf{U}, \mathsf{SK}_\mathsf{F}, \mathsf{VK}_\mathsf{F}, \pi) \leftarrow \mathsf{BiGen}(1^k) \\ (m,s) \leftarrow \mathsf{F}^{\Pi}(\mathsf{VK}_\mathsf{U}, \mathsf{VK}_\mathsf{F}, \mathsf{SK}_\mathsf{F}) \end{array}\right]$$

3. *$\mathcal{S}$ is* UF ***against any user*** U *for any PPT adversary* U, *if* $|\mathsf{Succ}_{\mathsf{U},\mathcal{S}}(1^k)|$ *is negligible, where* $\mathsf{Succ}_{\mathsf{U},\mathcal{S}}$ *is defined as below, and* U *is not allowed to ask the signature oracle for* $\mathsf{BiSig}(m)$.

$$\mathsf{Succ}_{\mathcal{A},\mathcal{S}} \ \stackrel{\text{def}}{=} \ \Pr\left[\ \mathsf{BiVer}(m,s) = succeed \ \middle|\ \begin{array}{l} (\mathsf{SK}_\mathsf{U}, \mathsf{VK}_\mathsf{U}, \mathsf{SK}_\mathsf{F}, \mathsf{VK}_\mathsf{F}, \pi) \leftarrow \mathsf{BiGen}(1^k) \\ (m,s) \leftarrow \mathcal{A}^{\mathsf{BiSig}_\mathsf{U}}(\mathsf{VK}_\mathsf{U}, \mathsf{VK}_\mathsf{F}) \end{array}\right]$$

Table 5. Bidirectional signature definitions.

7.1 Bidirectional Generic Signature Scheme

First, let's consider a standard signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$. The next paragraph explains how to build an bidirectional generic signature scheme $\mathcal{S}' = (\mathsf{BiGen}, \mathsf{BiSig}, \mathsf{BiVer}, \Pi)$ from the original scheme $\mathcal{S}$. The key generation algorithm BiGen uses the original key generation algorithm $\mathsf{Sig\text{-}Gen}$ to generate three keys (k_1, k_2, k_3), where $k_i = (\mathsf{SK}_i, \mathsf{VK}_i)$, and gives them to the users U, F, and P, such that they have in common only one key. For example, the user U gets $\mathsf{VK}_\mathsf{U} = (\mathsf{VK}_1, \mathsf{VK}_2)$, $\mathsf{SK}_\mathsf{U} = (\mathsf{SK}_1, \mathsf{SK}_2)$, the user F gets $\mathsf{VK}_\mathsf{F} = (\mathsf{VK}_1, \mathsf{VK}_3)$, $\mathsf{SK}_\mathsf{F} = (\mathsf{SK}_1, \mathsf{SK}_3)$, and the proxy P gets $\mathsf{VK}_\mathsf{P} = (\mathsf{VK}_2, \mathsf{VK}_3), \mathsf{SK}_\mathsf{U} = (\mathsf{SK}_2, \mathsf{SK}_3)$. The signature algorithm BiSig computes the signature of a message $m \in M$ by applying the standard signature algorithm twice, once for each key. For example, the user U signs a message m as $(s_1, s_2) = \mathsf{Sig}_1(m)\mathsf{Sig}_2(m)$. BiVer verifies if a signature generated with BiSig is correct, by executing the standard verification algorithm twice $\mathsf{Ver}_1(s_1)$ and $\mathsf{Ver}_2(s_2)$. The proxy function Π transforms a valid signature generated by BiSig for a pair of keys into a valid signature generated with another pair of keys: $\Pi(\mathsf{BiSig}_\mathsf{U}(m)) = \mathsf{BiSig}_\mathsf{F}(m)$.

The generic bidirectional signature scheme is secure if the next theorem is true. The proofs are in Appendix D.1.

Theorem 9 *Let's consider a standard signature scheme* $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$. *Based on* $\mathcal{S}$, *we build an bidirectional signature scheme* $\mathcal{S}' = (\mathsf{BiGen}, \mathsf{BiSig}, \mathsf{BiVer}, \Pi)$. *If* $\mathcal{S}$ *is* UF *than* $\mathcal{S}'$ *is* UF *against (1) the proxy* P, *(2) the user* F, *and (3) all users* U.

8 Unidirectional and Bidirectional Private Key Primitives

Unidirectional and bidirectional private key schemes for encrypting and signing messages (MAC) can be easily build on top of pseudo-random functions (PRF). Thus, it suffices to build unidirectional and bidirectional PRF functions. The encryption can be defined as $\mathsf{Enc}(m) = \langle r, f(r) \oplus m \rangle$, where r is chosen at random, and f is a PRF. The signature is defined as $\mathsf{Sig}(m) = f(m)$, which in fact is message authetication code (MAC).

8.1 Unidirectional PRF Functions

Informally, a unidirectional PRF function allows two users P and F to compute its value in any given point x, even if none of them knows the entire description of the function.

Definition 9 *A PRF function f is a unidirectional PRF function if there exist f_1 and f_2 two PRF functions such that the value $f(x)$ can be computed as $f_1(f_2(x))$.*

Based on the above definition, we construct the following unidirectional PRF. Let's consider $F = f_s$ a family of pseudo random function with seed s. We define UniPRF such that the value of the unidirectional PRF in a given point x is defined as $\mathsf{UniPRF}(x) = g_{s_1}(x) \oplus g_{s_2}(x)$, where $g \in F$. The proxy P and the user F are both given one of the two seeds. For example, the proxy P receives s_1 and the user F gets s_2. In this way, they can cooperate and compute $\mathsf{UniPRF}(x)$ by first computing $g_{s_1}(x)$ and $g_{s_2}(x)$ and applying the XOR operation.

Theorem 10 *The unidirectional PRF function defined as $\mathsf{UniPRF}(x) = f_1(x) \oplus f_2(x)$, where f_1 and f_2 are two PRF functions, is a PRF function.*

8.2 Bidirectional PRF Functions

In a similar way, we can informally define a bidirectional PRF function as a PRF function that can be transformed into a new PRF function. This means that one can compute the value $f_2(x)$, given an initial value $f_1(x)$ and a simple transformation Π.

Definition 10 *A PRF function f_1 is a bidirectional PRF function if for any PRF function f_2, there exists a transformation Π such that the value $f_2(x)$ can be computed as $\Pi(f_1(x))$.*

Let's consider $F = f_s$ a family of PRF functions with seed s. We construct a PRF function $\mathsf{BiPRF}_1(x) = g_{s_1}(x) \oplus g_{s_2}(x)$, where $g \in F$ is a PRF function. The function BiPRF is bidirectional because from any value $\mathsf{BiPRF}_1(x)$, one can easily obtain the value of $\mathsf{BiPRF}_2(x) = g_{s_1}(x) \oplus g_{s_3}(x)$ by computing $\mathsf{BiPRF}_1(x) \oplus (g_{s_2}(x) \oplus g_{s_3}(x))$. The transformation function is defined as $\Pi(x) = g_{s_2}(x) \oplus g_{s_3}(x)$. The obvious way to split the keys between F, P, and $user$ is to give 2 keys to each one such that any two users have only 1 key in common. For example, the user U gets (s_1, s_2), the proxy P gets (s_2, s_3), and the user F receives (s_1, s_3).

Theorem 11 *The bidirectional PRF function defined as $\mathsf{BiPRF}(x) = f_1(x) \oplus f_2(x)$, where f_1 and f_2 are two PRF functions, is a PRF function.*

9 Conclusions

At the beginning of the paper we started describing the for *unidirectional* functions. The unidirectional notation is justified by the fact that the proxy P needs to help the law enforcement agency (user F) for every message that needs to be decrypted or signed. There is also an offline version of the *unidirectional* proxy function. Because of space consideration, we shortly describe it as part of the conclusions. The offline proxy functions are based on the key-insulated encryption and signature primitives created by [9, 10]. In the offline scheme, users U protect their secrets by periodically updating their secret keys. The user U updates its secret key using the index of the current time period and some information provided by a third party (P). The same third party P can help the law enforcement agency F to decrypt or sign messages on behalf of other users U, by providing special pieces of information at the beginning of the time period. There is one main difference between unidirectional and offline proxy functions: in the unidirectional case, the law enforcement agency F is strictly controlled by the proxy P and cannot decrypt or sign a message without its help; in the offline case, the law enforcement agency can misbehave [3] for one time period once it has the necessary information.

The main contribution of this paper is the formalized definitions of the *bidirectional* and *unidirectional* proxy functions for encryption and signatures and their security guarantees. In addition, for each class of proxy functions, the paper describes one generic technique and several specific techniques to transform a standard cryptographic primitive into a proxy function.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[2] M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. *Lecture Notes in Computer Science*, 1070, 1996.

[3] M. Bellare and R. Sandhu. The Security of Practical Two-party RSA Signature Schemes. *Cryptology ePrint Archive, Report 2001/060*, 2001.

[4] M. Blaze and M. Strauss. Atomic Proxy Cryptography. *Eurocrypt*, 1998.

[5] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Proceedings of Crypto '2001, Lecture Notes in Computer Science*, 2139:213–229, 2001.

---

[3] Misbehave i.e. decrypt or sign messages it is not supposed to

[6] C. Boyd. *Digital Multisignatures*, volume Cryptography and Coding, pages 241–246. Claredon Press, 1986.

[7] J.-S. Coron. On the Exact Security of Full Domain Hash. *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference*, pages 229–235, 2000.

[8] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. *Advances in Cryptology – Crypto '89*, pages 307–315, 1989.

[9] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public Key Cryptosystems. *Eurocrypt*, 2002.

[10] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. *PKC*, 2002.

[11] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. *CRYPTO*, 263:186–194, 1987.

[12] T. E. Gamal. A Public Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm. *IEEE Transactions of Information Theory*, pages 31(4): 469–472, 1985.

[13] B. Horne, B. Pinkas, and T. Sander. Escrow Services and Incentives in Peer-to-Peer Networks. *3rd ACM Conference on Electronic Commerce*, 2001.

[14] J. Kilian and F. T. Leighton. Fair Cryptosystems, Revisited. *Advances of Cryptology - CRYPTO '95 Proceedings, Berlin: Springer-Verlag*, 1995.

[15] H. Kim, J. Baek, B. Lee, and K. Kim. Computing with Secrets for Mobile Agent Using One-time Proxy Signature. *SCIS2001, vol 2/2*, pages 845–850, 2001.

[16] B. Lee, H. Kim, and K. Kim. Strong Proxy Signature and its Applications. *SCIS2001, vol 2/2*, pages 603–608, 2001.

[17] F. T. Leighton. Failsafe Key Escrow Systems. *Technical Memo 483, MIT Lab. for Computer Science*, 1994.

[18] A. K. Lenstra, P. Winkler, and Y. Yacobi. A Key Escrow System with Warrant Bounds. *Advances in Cryptology - CRYPTO*, pages 197–207, 1995.

[19] P. MacKenzie. An Efficient Two-Party Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. *PKC*, 2003.

[20] P. MacKenzie and M. Reiter. Delegation of cryptographic servers for capture-resilient devices. *CCS*, 2001.

[21] P. MacKenzie and M. K. Reiter. Networked Cryptographic Devices Resilient to Capture. *Eighth ACM Conference on Computer and Communications Security (CCS-8)*, 2001.

[22] P. MacKenzie and M. K. Reiter. Two-Party Generation of DSA Signatures. *Advances in Cryptology - CRYPTO 2001 (Lecture Notes in Computer Science 2139)*, 2001.

[23] S. Micali. Fair Public-Key Cryptosystems. *Advances in Cryptology - CRYPTO '92 Proceedings, Berlin: Springer-Verlag*, 1993.

[24] A. Nicolosi, M. Krohn, Y. Dodis, and D. Mazieres. Proactive Signatures for User Authentication. *NDSS*, 2003.

[25] R. L. Rivest, A. Shamir, and L. M. Adelman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82, 1977.

[26] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *Advances in Cryptology: Proceedings of CRYPTO 84, Lecture Notes in Computer Science*, 7:47–53, 1984.

[27] Y.Frankel and M.Yung. Escrow Encryption Systems Visited: Attacks, Analysis and Designs. *Advances in Cryptology - CRYPTO '95 Proceedings, Berlin:Springer-Verlag*, 1995.

A Unidirectional Encryption Scheme

A.1 Unidirectional Generic Encryption Scheme

Theorem 1 *Let's consider a standard encryption scheme $\mathcal{E} = (\text{Enc-Gen}, \text{Enc}, \text{Dec})$. Based on $\mathcal{E}$, we build an unidirectional encryption scheme $\mathcal{E}' = (\text{UniGen}, \text{UniEnc}, \text{UniDec}, \text{PDec}, \text{FDec})$. If $\mathcal{E}$ is CCA2 secure, than $\mathcal{E}'$ is also CCA2 secure against (1) the proxy P, (2) the user F, and (3) all users U, where the success of the adversary is defined in table 6.*

Proof

1. Let's assume that $\mathcal{E}'$ is not CCA2 secure against the proxy P. This means that the proxy P can break $\mathcal{E}'$ with probability of success greater than 1/2. We assume that PDec is a deterministic algorithm and the proxy P never submits $\text{PDec}(\text{UniEnc}_{\text{EK}}(m_b))$ to the FDec oracle. Based on P, we build an adversary $\mathcal{B}$ capable of breaking the original encryption scheme $\mathcal{E}$. The adversary $\mathcal{B}$ receives as input the public key EK_2 of the original encryption scheme $\mathcal{E}$. $\mathcal{B}$ simulates the conditions necessary for the proxy P to break the unidirectional encryption $\mathcal{E}'$ by randomly choosing a public/private key pair $(\text{EK}_1, \text{DK}_1)$ and forwarding it to the proxy P together with EK_2. The adversary $\mathcal{B}$ starts running the proxy P. Whenever the proxy P needs to make a query to the FDec oracle, $\mathcal{B}$ simulates the FDec oracle by taking the P's query q, and forwards it to its own Dec oracle access. $\mathcal{B}$ sends to the proxy P the output of the Dec oracle. At one moment, P challenges the unidirectional encryption scheme $\mathcal{E}'$ by choosing two messages (m_0, m_1) and sending them to $\mathcal{B}$. $\mathcal{B}$ chooses the same two messages to challenge the original encryption scheme $\mathcal{E}$. When $\mathcal{B}$ is presented with the challenge $\text{Enc}_2(m_b)$, where m_b is chosen at random from the two messages (m_0, m_1), $\mathcal{B}$ applies $\text{Enc}_1(\text{Enc}_2(m))$ and sends the challenge as $\text{UniEnc}(m_b)$ to the proxy P. We assumed that the proxy P can break the unidirectional encryption scheme with probability greater than 1/2. Thus, $\mathcal{B}$ can break the standard encryption scheme with probability greater than 1/2.

2. Let's assume that $\mathcal{E}'$ is not CCA2 secure against the user F. This means that the user F can break $\mathcal{E}'$ with probability of success greater than 1/2. Based on the user F, we build an adversary $\mathcal{B}$ capable of breaking the orginal encryption scheme $\mathcal{E}$. The adversary $\mathcal{B}$ receives as input the public key EK_1 of the original

Generic	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(\mathsf{EK}_1,\mathsf{DK}_1,\mathsf{EK}_2,\mathsf{DK}_2)\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{P}^{\mathsf{FDec}}(\mathsf{EK}_1,\mathsf{EK}_2,\mathsf{DK}_1),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{P}^{\mathsf{FDec}}(\mathsf{EK}_1,\mathsf{EK}_2,\mathsf{DK}_1,\mathsf{UniEnc}(m_b))\end{array}\right]$ $\mathsf{Succ}_{\mathsf{F},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(\mathsf{EK}_1,\mathsf{DK}_1,\mathsf{EK}_2,\mathsf{DK}_2)\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{F}^{\mathsf{PDec}}(\mathsf{EK}_1,\mathsf{EK}_2,\mathsf{DK}_2),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{F}^{\mathsf{PDec}}(\mathsf{EK}_1,\mathsf{EK}_2,\mathsf{DK}_2,\mathsf{UniEnc}(m_b))\end{array}\right]$
El-Gamal	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(\mathsf{EK},\mathsf{DK})\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{P}(\mathsf{EK},\mathsf{DK}_\mathsf{P}),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{P}(\mathsf{EK},\mathsf{DK}_\mathsf{P},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\right]$ $\mathsf{Succ}_{\mathsf{F},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(\mathsf{EK},\mathsf{DK})\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{F}^{\mathsf{PDec}*}(\mathsf{EK},\mathsf{DK}_\mathsf{F}),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{F}^{\mathsf{PDec}*}(\mathsf{EK},\mathsf{DK}_\mathsf{F},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\right]$
RSA	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}} \stackrel{\text{def}}{=} \Pr\left[m'=m \,\middle|\, (\mathsf{EK},\mathsf{DK})\leftarrow\mathsf{UniGen}(1^k),m\leftarrow M,m'\leftarrow\mathsf{P}(\mathsf{EK},\mathsf{DK}_\mathsf{P},\mathsf{UniEnc}_{\mathsf{EK}}(m))\right]$
IBE	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(s=(s_1,s_2),sP)\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{P}(s_1,sP),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{P}(s_1,sP,\mathsf{UniEnc}_{ID}(m_b))\end{array}\right]$ $\mathsf{Succ}_{\mathsf{F},\mathcal{E}'} \stackrel{\text{def}}{=} \Pr\left[b=\tilde{b} \,\middle|\, \begin{array}{c}(s=(s_1,s_2),sP)\leftarrow\mathsf{UniGen}(1^k),(m_0,m_1)\leftarrow\mathsf{F}^{\mathsf{PDec}*}(\mathsf{EK},\mathsf{DK}_\mathsf{F}),\\ b\leftarrow\{0,1\},\tilde{b}\leftarrow\mathsf{F}^{\mathsf{PDec}*}(\mathsf{EK},\mathsf{DK}_\mathsf{F},\mathsf{UniEnc}_{\mathsf{EK}}(m_b))\end{array}\right]$

Table 6. Online generic encryption definitions for adversary's success.

encryption scheme $\mathcal{E}$. First, $\mathcal{B}$ simulates the conditions necessary for the user F to break the unidirectional encryption $\mathcal{E}'$ by choosing at random a public/private key pair $(\mathsf{EK}_2,\mathsf{DK}_2)$ and forwarding it to F together with the EK_1. The adversary $\mathcal{B}$ starts running the user F. When the user F makes a query $q=\mathsf{Enc}_1(\mathsf{Enc}_2(m))$ to the PDec oracle, $\mathcal{B}$ takes the query e, forwards it to its own Dec oracle, and sends the answer $\mathsf{Enc}_2(m)$ directly to F. When the user F challenges the adversary $\mathcal{B}$, it chooses two messages (m_0,m_1) and sends them to $\mathcal{B}$. $\mathcal{B}$ encrypts those two messages using the key EK_2 and sends them to challenge the standard encryption. When $\mathcal{B}$ is presented with the challenge $\mathsf{Enc}_1(m_b)$, where m_b is chosen at random from two messages $(\mathsf{Enc}_2(m_0),\mathsf{Enc}_2(m_1))$, $\mathcal{B}$ sends the challenge to F. We assumed that the user F can break the unidirectional encryption scheme with probability greater than 1/2. Thus, $\mathcal{B}$ can break the standard encryption scheme with probability greater than 1/2.

3. This part is implied by the two previous parts.

A.2 Unidirectional El Gamal Encryption Scheme

Theorem 2 *Let* $\mathcal{E}' = (\mathsf{UniGen},\mathsf{UniEnc},\mathsf{UniDec},\mathsf{PDec},\mathsf{FDec})$ *be an unidirectional El Gamal encryption scheme.* $\mathcal{E}'$ *is* CPA *secure against (1) the proxy* P*, (2) the user* F*, and (3) all users* U*, where the adversary's success is defined in table 6.*

Proof

1. Let's assume that P is capable of breaking the unidirectional El-Gamal encryption scheme. This means that $|\mathsf{Succ}_{\mathsf{P},\mathcal{E}'}(1^k)-1/2|$ is not negligible. Based on P, we will build an adversary $\mathcal{B}$ that breaks the original El-Gamal scheme with non-negligible probability. Initially, $\mathcal{B}$ is given the public key EK of the original EL-Gamal scheme. Based on EK, $\mathcal{B}$ simulates the conditions necessary for P to break the unidirectional ElGamal. For this, $\mathcal{B}$ chooses a random number as x_1 and gives it to P. As part of the unidirectional challenge, P chooses two messages (m_0,m_1) and forwards them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard El-Gamal. When $\mathcal{B}$ is given the encryption of m_b, where $b\leftarrow\{0,1\}$, it forwards $\mathsf{Enc}(m_b)$ to P. We assumed that P is capable of breaking the unidirectional El-Gamal and by definition, $\mathsf{UniEnc}(m_b)=\mathsf{Enc}(m_b)$. Thus, if P decrypts

UniEnc(m_b) with probability greater than 1/2, then $\mathcal{B}$ decrypts Enc(m_b) with probability greater than 1/2.

2. Let's assume that F is capable of breaking the unidirectional El-Gamal encryption scheme. This means that $\mid \mathsf{Succ}_{\mathsf{F},\mathcal{E}'}(1^k) - 1/2 \mid$ is not negligible. We use the notation $\mathsf{F}^{\mathsf{P}*}$ to indicate that the user F can have honest access to the PDec function. Based on F, we will build an adversary $\mathcal{B}$ that breaks the original El-Gamal scheme with non-negligible probability. Initially, $\mathcal{B}$ is given the public key EK of the original EL-Gamal scheme. Based on EK, $\mathcal{B}$ simulates the conditions necessary for F to break the unidirectional ElGamal. For this, $\mathcal{B}$ chooses a random number as x_2 and gives it to F. The adversary $\mathcal{B}$ simulates the honest access of F to PDec and the encryption oracle UniEnc by taking the query message m and returning mg^{ex_2} and Enc(m). As part of the unidirectional challenge, F chooses two messages (m_0, m_1) and forwards them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard El-Gamal. When $\mathcal{B}$ is given the encryption of m_b, where $b \leftarrow \{0, 1\}$, it forwards Enc(m_b) to F. We assumed that F is capable of breaking the unidirectional El-Gamal and by definition, UniEnc(m_b) = Enc(m_b). Thus, if F decrypts UniEnc(m_b) with probability greater than 1/2, then $\mathcal{B}$ decrypts Enc(m_b) with probability greater than 1/2.

3. This part is implied by the two previous parts.

A.3 Unidirectional RSA Encryption Scheme

Definition 11 *A function* $f : \{0,1\}^* \rightarrow \{0,1\}^*$ *is* ONE-WAY *if it satisfies two conditions:*

1. *There exists a* poly-*time algorithm that correctly computes* $f(x)$ *for any* $x \in \{0, 1\}^*$.

2. *For any PPT adversary* $\mathcal{A}$, $\Pr(f(z) = y \mid x \leftarrow \{0,1\}^k; y = f(x); z \leftarrow \mathcal{A}(y, 1^k)) \leq negl(k)$.

Theorem 3 *Let* $\mathcal{E}'$ = (UniGen, UniEnc, UniDec, PDec, FDec) *be an unidirectional RSA encryption scheme.* $\mathcal{E}'$ *is* ONE-WAY *secure against (1) the proxy* P, *(2) the user* F, *and (3) all users* U, *where the success of the adversary is defined in table 6.*

Proof

1. Let's assume that P breaks the unidirectional RSA. This means that $\mathsf{Succ}_{\mathsf{P},\mathcal{E}}(1^k)$ is not negligible. We

will show that based on P we can build an adversary $\mathcal{B}$ that breaks the original RSA encryption scheme. $\mathcal{B}$ is given the public key EK and based on it creates the conditions necessary for P to break the unidirectional RSA. $\mathcal{B}$ chooses a random number d_1 and forwards it to P as its part of the secret key. The goal of the adversary $\mathcal{B}$ is to find m' such that $m' = m$ when given Enc(m). $\mathcal{B}$ forwards the ciphertext Enc(m) = UniEnc(m) to P. We assumed that P is able to find $m' = m$ when given UniEnc(m) with non-negligible probability. Thus, $\mathcal{B}$ is able to find m'.

2. This proof is similar to the previous one.

3. This part is implied by the two previous parts.

A.4 Unidirectional IBE Encryption Scheme

Theorem 4 *Let* $\mathcal{E}'$ = (UniGen, UniEnc, UniDec, PDec, FDec) *be an unidirectional IBE encryption scheme.* $\mathcal{E}'$ *is* CPA *secure against (1) the proxy* P, *(2) the user* F, *and (3) all users* U, *where the success of the adversary is defined in table 6.*

Proof

1. Let's assume that P is capable of breaking the unidirectional IBE encryption scheme. This means that $\mid \mathsf{Succ}_{\mathsf{P},\mathcal{E}'}(1^k) - 1/2 \mid$ is not negligible. Based on P, we will build an adversary $\mathcal{B}$ that breaks the original IBE scheme with non-negligible probability. Initially, $\mathcal{B}$ is given the public key $P_{pub} = sP$ of the original IBE scheme. Based on sP, $\mathcal{B}$ simulates the conditions necessary for P to break the unidirectional IBE. For this, $\mathcal{B}$ chooses a random number as s_1 and gives it to P. As part of the unidirectional challenge, P chooses two messages (m_0, m_1) and forwards them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard IBE. When $\mathcal{B}$ is given the encryption of m_b, where $b \leftarrow \{0, 1\}$, it forwards Enc(m_b) to P. We assumed that P is capable of breaking the unidirectional IBE and by definition, UniEnc(m_b) = Enc(m_b). Thus, if P decrypts UniEnc(m_b) with probability greater than 1/2, then $\mathcal{B}$ decrypts Enc(m_b) with probability greater than 1/2.

2. Let's assume that F is capable of breaking the unidirectional IBE encryption scheme. This means that $\mid \mathsf{Succ}_{\mathsf{F},\mathcal{E}'}(1^k) - 1/2 \mid$ is not negligible. We use the notation $\mathsf{F}^{\mathsf{P}*}$ to indicate that the user F can have honest access to the PDec function. Based on F, we

will build an adversary $\mathcal{B}$ that breaks the original IBE scheme with non-negligible probability. Initially, $\mathcal{B}$ is given the public key $P_{pub} = sP$ of the original IBE scheme. $\mathcal{B}$ simulates the conditions necessary for F to break the unidirectional IBE by choosing a random number as s_2 and giving it to F. The adversary $\mathcal{B}$ simulates the honest access of F to PDec and the encryption oracle UniEnc by taking the query message m and returning $\hat{e}(rID, s_1P)$ and $\mathsf{Enc}(m) = < rID, m \oplus \hat{e}(rID, sP) >$. As part of the unidirectional challenge, F chooses two messages (m_0, m_1) and forwards them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard IBE. When $\mathcal{B}$ is given the encryption of m_b, where $b \leftarrow \{0, 1\}$, it forwards $\mathsf{Enc}(m_b)$ to F. We assumed that F is capable of breaking the unidirectional El-Gamal and by definition, $\mathsf{UniEnc}(m_b) = \mathsf{Enc}(m_b)$. Thus, if F decrypts $\mathsf{UniEnc}(m_b)$ with probability greater than 1/2, then $\mathcal{B}$ decrypts $\mathsf{Enc}(m_b)$ with probability greater than 1/2.

3. This part is implied by the two previous parts.

B Unidirectional Signature Scheme

B.1 Unidirectional Generic Signature Scheme

Theorem 5 *Let* $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a standard signature scheme. Let's consider* $\mathcal{S}' = (\mathsf{UniGen}, \mathsf{UniSig}, \mathsf{UniVer}, \mathsf{PSig}, \mathsf{FSig})$ *an unidirectional signature scheme constructed as described above, based on* $\mathcal{S}$*. If* $\mathcal{S}$ *is* UF-CMA *, than* $\mathcal{S}'$ *is* UF-CMA *against (1) the proxy* P*, (2) the user* F*, and (3) all users* U*, where the adversary's success is defined in table 7.*

Proof

1. Let's assume that $\mathcal{S}'$ is not UF-CMA against the proxy P. This means that $|\mathsf{Succ}_{P,\mathcal{S}'}(1^k)|$ is not negligible. We assume that the proxy P is not allowed to ask the FSig oracle for $\mathsf{FSig}(m)$. Based on $\mathcal{S}'$ we build a forger $\mathcal{B}$ capable to break the original signature scheme $\mathcal{S}$. The forger $\mathcal{B}$ receives as input the public key VK_2 and tries to generate a valid signature of a message m under the secret key SK_2. The forger $\mathcal{B}$ chooses at random a public/private key pair $(\mathsf{VK}_1, \mathsf{SK}_1)$ and forwards it to P together with VK_2. The forger $\mathcal{B}$ starts running the proxy P. When P makes a query on the hash oracle for a message m', the forger $\mathcal{B}$ forwards the request to its own hash oracle and sends the answer to the proxy P. When P

asks the FSig oracle to produce a signature for a message m', $\mathcal{B}$ asks its own signature oracle to produce a signature for m' under SK_2 and sends the result to the proxy P. At one moment, the proxy P generates a valid unidirectional signature for a message m with a non-negligible probability, where m is a completely new message. $\mathcal{B}$ takes the unidirectional signature $\mathsf{UniSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_2(m)$, removes the first part and outputs $\mathsf{Sig}_2(m)$ as a valid signature of m.

2. Let's assume that $\mathcal{S}'$ is not UF-CMA against F. This means that $|\mathsf{Succ}_{F,\mathcal{S}'}(1^k)|$ is not negligible. We assume that F is not allowed to ask the PSig oracle about m. Based on $\mathcal{S}'$ we build a forger $\mathcal{B}$ capable to break the original signature scheme $\mathcal{S}$. The forger $\mathcal{B}$ receives as input the public key VK_1 and tries to generate a valid signature of a message m under the secret key SK_1. The forger $\mathcal{B}$ chooses at random a public/private key pair $(\mathsf{VK}_2, \mathsf{SK}_2)$ and forwards it to the user F together with VK_1. The forger $\mathcal{B}$ starts running F. When the user F makes a query on the hash oracle for a message m', the forger $\mathcal{B}$ forwards the request to its own hash oracle and sends the answer back to F. When F asks the PSig oracle to produce part of the signature for a message m', $\mathcal{B}$ asks its own signature oracle to produce a signature for m' under SK_1. After that, $\mathcal{B}$ sends to F $\mathsf{Sig}_1(m')$. At one moment, F generates a valid unidirectional signature for a message m with a non-negligible probability, where m is a completely new message. $\mathcal{B}$ takes the unidirectional signature $\mathsf{UniSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_2(m)$, removes the second part and outputs $\mathsf{Sig}_1(m)$ as a valid signature of m.

3. This part is implied by the previous two parts.

B.2 Unidirectional RSA-Hash Signature Scheme

Theorem 6 *Let* $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a classic RSA-Hash signature scheme. Let's consider* $\mathcal{S}' = (\mathsf{UniGen}, \mathsf{UniSig}, \mathsf{UniVer}, \mathsf{PSig}, \mathsf{FSig})$ *an unidirectional RSA-Hash signature scheme constructed as described above.* $\mathcal{S}'$ *is* UF-CMA *against (1) the proxy* P*, (2) the user* F*, and (3) all users* U*, where the adversary's success is defined in table 7.*

Proof

1. Let's assume that P can break the unidirectional RSA-Hash scheme. This means that $|\mathsf{Succ}_{P,\mathcal{S}}(1^k)|$ is not negligible. Based on the proxy P, we build a forger $\mathcal{B}$

Generic	$\mathsf{Succ}_{\mathsf{P},\mathcal{S}'} \overset{\text{def}}{=} \Pr\left[\mathsf{UniVer}(m,s)=succeed \;\middle	\; \begin{array}{c}(\mathsf{VK}_1,\mathsf{SK}_1,\mathsf{VK}_2,\mathsf{SK}_2)\leftarrow \mathsf{UniGen}(1^k)\\ (m,s)\leftarrow \mathsf{P}^{\mathsf{FSig}}(\mathsf{VK}_1,\mathsf{VK}_2,\mathsf{SK}_1)\end{array}\right]$
	$\mathsf{Succ}_{\mathsf{F},\mathcal{S}'} \overset{\text{def}}{=} \Pr\left[\mathsf{UniVer}(m,s)=succeed \;\middle	\; \begin{array}{c}(\mathsf{VK}_1,\mathsf{SK}_1,\mathsf{VK}_2,\mathsf{SK}_2)\leftarrow \mathsf{UniGen}(1^k)\\ (m,s)\leftarrow \mathsf{F}^{\mathsf{PSig}}(\mathsf{VK}_1,\mathsf{VK}_2,\mathsf{SK}_2)\end{array}\right]$
RSA-Hash	$\mathsf{Succ}_{\mathsf{P},\mathcal{S}} \overset{\text{def}}{=} \Pr\left[\mathsf{UniVer}(m,s)=succeed \;\middle	\; (\mathsf{SK},\mathsf{VK})\leftarrow \mathsf{UniGen}(1^k),(m,s)\leftarrow \mathsf{P}^{\mathsf{FSig}}(\mathsf{SK_P},\mathsf{VK})]\right]$
	$\mathsf{Succ}_{\mathsf{F},\mathcal{S}} \overset{\text{def}}{=} \Pr\left[\mathsf{UniVer}(m,s)=succeed \;\middle	\; (\mathsf{SK},\mathsf{VK})\leftarrow \mathsf{UniGen}(1^k),(m,s)\leftarrow \mathsf{F}^{\mathsf{PSig}}(\mathsf{SK_F},\mathsf{VK})\right]$

Table 7. Online generic signature definitions for adversary's success.

capable of breaking the RSA encryption scheme. The forger $\mathcal{B}$ receives as input a public key (N,e) and tries to invert $x = f^{-1}(y)$, where f is the RSA function defined by N and e. The adversary $\mathcal{B}$ starts running the proxy P for this public key and a randomly chosen number d_1 given as a secret key. When the proxy P makes the i-th hash query, the adversary looks to see if the message m_i was already asked. If not, it picks a random x_i, sets $h(m_i) = x_i^e$ with probability p and $h(m_i) = y * x_i^e$ with probability $1 - p$. If the proxy P makes a query to FSig for a message m_i, the adversary returns x_i if m_i was asked before. Otherwise it fails. Eventually, P outputs a correct unidirectional RSA-Hash signature (m,s) for a brand new message m. If the message m was not hashed before, the adversary computes its has value. If $h(m) = y * x_i^e$, then the adversary returns $y^d = s/x_i^e$ as the $x = f^{-1}(y)$. Otherwise, it fails.

2. Let's assume that F can break the unidirectional RSA-Hash scheme. This means that $|\mathsf{Succ}_{\mathsf{F},\mathcal{S}}(1^k)|$ is not negligible. Based on the user F, we build a forger $\mathcal{B}$ capable of breaking the RSA encryption scheme. The forger $\mathcal{B}$ receives as input a public key (N,e) and tries to invert $x = f^{-1}(y)$, where f is the RSA function defined by N and e. The adversary $\mathcal{B}$ starts running the user F for this public key and a randomly chosen number d_2 given as a secret key. When the user F makes the i-th hash query, the adversary looks to see if the message m_i was already asked. If not, it picks a random x_i, sets $h(m_i) = x_i^e$ with probability p and $h(m_i) = y * x_i^e$ with probability $1 - p$. If the user F makes a query to PSig for a message m_i, the adversary returns x_i if m_i was asked before. Otherwise it fails. Eventually, F outputs a correct unidirectional RSA-Hash signature (m,s) for a brand new message m. If the message m was not hashed before, the ad-

versary computes its has value. If $h(m) = y * x_i^e$, then the adversary returns $y^d = s/x_i^e$ as the $x = f^{-1}(y)$. Otherwise, it fails.

3. This part is implied by the previous two parts.

C Bidirectional Encryption Scheme

C.1 Bidirectional Generic Encryption Scheme

Theorem 7 *Let's consider a standard encryption scheme* $\mathcal{E} = (\mathsf{Enc\text{-}Gen}, \mathsf{Enc}, \mathsf{Dec})$. *Based on* $\mathcal{E}$, *we build an bidirectional encryption scheme* $\mathcal{E}' = (\mathsf{BiGen}, \mathsf{BiEnc}, \mathsf{BiDec}, \mathsf{PDec}, \mathsf{FDec})$. *If* $\mathcal{E}$ *is* CCA2 *secure, than* $\mathcal{E}'$ *is also* CCA2 *secure against (1) the proxy* P, *(2) the user* F, *and (3) all users* U, *where the adversary's success is defined in table 8.*

For technical reasons, we assume that there exists an efficient algorithm that evaluates the relation $R_\pi(e,e')$ to true or false, where $e = \mathsf{BiEnc}(m)$ is the original chiphertext and $e' = \Pi(e)$ is the modified ciphertext computed by the proxy P. The output of the algorithm is true, it must be the case that $\mathsf{Dec}_{\mathsf{EK_U}}(e) = \mathsf{Dec}_{\mathsf{EK_F}}(e')$. Having such an algorithm, we allow the proxy P has oracle access to BiDec because it can never submit a ciphertext e' such that $R_\pi(e,e') = true$.

Proof

1. Let's assume that $\mathcal{E}'$ is not CCA2 secure against P. This means that $|\mathsf{Succ}(\mathsf{P},\mathcal{E}') - 1/2|$ is not negligible. Based on P, we will build an adversary $\mathcal{B}$ that breaks the standard encryption scheme $\mathcal{E}$ for the key $k_2 = (\mathsf{EK}_2,\mathsf{DK}_2)$. The adversary $\mathcal{B}$ tries to decrypt

Generic	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathsf{P}^{\mathsf{BiDec}}(\mathsf{EK_U}, \mathsf{EK_F}, \pi), \tilde{b} \leftarrow \mathsf{P}^{\mathsf{BiDec}}(\mathsf{EK_U}, \mathsf{EK_F}, \pi, \mathsf{BiEnc}(m_b)) \end{array} \right]$ $\mathsf{Succ}_{\mathsf{F},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathsf{F}^{\mathsf{P}}(\mathsf{EK_U}, \mathsf{EK_F}, \mathsf{DK_F}), \tilde{b} \leftarrow \mathsf{F}^{\mathsf{P}}(\mathsf{EK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \mathsf{BiEnc}(m_b)) \end{array} \right]$ $\mathsf{Succ}_{\mathcal{A},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{BiDec}}(\mathsf{EK_U}, \mathsf{EK_F}), \tilde{b} \leftarrow \mathcal{A}^{\mathsf{BiDec}}(\mathsf{EK_U}, \mathsf{EK_F}, \mathsf{BiEnc}(m_b)) \end{array} \right]$
El Gamal	$\mathsf{Succ}_{\mathsf{P},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathsf{P}(\mathsf{EK_U}, \mathsf{EK_F}, \pi), \tilde{b} \leftarrow \mathsf{P}(\mathsf{EK_U}, \mathsf{EK_F}, \pi, \mathsf{BiEnc}_{\mathsf{EK_U}}(m_b)) \end{array} \right]$ $\mathsf{Succ}_{\mathsf{F},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathsf{F}(\mathsf{EK_U}, \mathsf{EK_F}, \mathsf{DK_F}), \tilde{b} \leftarrow \mathsf{F}(\mathsf{EK_U}, \mathsf{EK_F}, \mathsf{DK_F}, \mathsf{BiEnc}_{\mathsf{EK_U}}(m_b)) \end{array} \right]$ $\mathsf{Succ}_{\mathcal{A},\mathcal{E}} \overset{\mathrm{def}}{=} \Pr\left[b = \tilde{b} \,\middle|\, \begin{array}{c} (\mathsf{EK_U}, \mathsf{DK_U}, \mathsf{EK_{\mathcal{A}}}, \mathsf{DK_{\mathcal{A}}}, \pi) \leftarrow \mathsf{BiGen}(1^k), b \leftarrow \{0,1\}, \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{EK_U}, \mathsf{EK_{\mathcal{A}}}, \mathsf{DK_{\mathcal{A}}}), \tilde{b} \leftarrow \mathcal{A}(\mathsf{EK_U}, \mathsf{EK_{\mathcal{A}}}, \mathsf{DK_{\mathcal{A}}}, \mathsf{BiEnc}_{\mathsf{EK_U}}(m_b)) \end{array} \right]$

Table 8. Bidirectional encryption definitions for adversary's success.

the ciphertext $\mathsf{Enc}_2(m)$. $\mathcal{B}$ chooses two pairs of keys (k_1, k_3), gives them to P, and then starts P. $\mathcal{B}$ simulates P's access to the decryption oracle BiDec by taking each query e of P, and sending $\mathsf{Dec}_{k_1}(e)$ to its own decryption oracle. The message received as the answer is sent to the proxy $]proxy$. P chooses two messages (m_0, m_1) to challenge the bidirectional encryption scheme $\mathcal{E}'$ and sends them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard encryption scheme $\mathcal{E}$. When $\mathcal{B}$ is presented with the challenge $\mathsf{Enc}_{k_2}(m_b)$, where $m_b \in (m_0, m_1)$, $\mathcal{B}$ sends to P $\mathsf{BiEnc}(m_b) = \mathsf{Enc}_{k_1}(\mathsf{Enc}_{k_2}(m_b))$. We assumed that P is able to break the bidirectional encryption scheme with non-negligible probability. Thus, $\mathcal{B}$ breaks the standard encryption scheme with non-negligible probability.

2. Let's assume that $\mathcal{E}'$ is not **CCA2** secure against F. This means that $|\mathsf{Succ}(\mathsf{F}, \mathcal{E}') - 1/2|$ is not negligible. Based on P, we will build an algorithm $\mathcal{B}$ that breaks the standard encryption scheme $\mathcal{E}$ for encryption key k_1. $\mathcal{B}$ chooses two random numbers as keys (k_2, k_3) and gives them to F. In addition, $\mathcal{B}$ simulates oracle access to P by taking each query e of F, forwarding it to its own decryption oracle and sending back to F $\mathsf{Enc}_{k_3}(\mathsf{Dec}_{k_1}(e))$. F chooses two messages (m_0, m_1) to challenge the bidirectional encryption scheme $\mathcal{E}'$ and sends them to $\mathcal{B}$. $\mathcal{B}$ encrypts the two messages and uses $(\mathsf{Enc}_{k_2}(m_0), \mathsf{Enc}_{k_2}(m_1))$ to

challenge the standard encryption scheme $\mathcal{E}$. When $\mathcal{B}$ is presented with the challenge $\mathsf{Enc}_{k_1}(m_b)$, where $m_b \in (\mathsf{Enc}_{k_2}(m_0), \mathsf{Enc}_{k_2}(m_1))$, $\mathcal{B}$ sends to F the challenge $\mathsf{Enc}_{k_1}(m_b)$. We assumed that F is able to break the bidirectional encryption scheme with non-negligible probability. Thus, $\mathcal{B}$ breaks the standard encryption scheme with non-negligible probability.

3. Let's assume that $\mathcal{E}'$ is not **CCA2** secure. This means that there is a PPT adversary $\mathcal{A}$ such that $|\mathsf{Succ}(\mathcal{A}, \mathcal{E}') - 1/2|$ is not negligible. Based on $\mathcal{A}$, we will build an adversary $\mathcal{B}$ that breaks the standard encryption scheme $\mathcal{E}$ for the key $(\mathsf{EK}_2, \mathsf{DK}_2)$. The adversary $\mathcal{B}$ tries to decrypt the ciphertext $\mathsf{Enc}_2(m)$. $\mathcal{B}$ starts $\mathcal{A}$. $\mathcal{B}$ simulates $\mathcal{A}$'s access to the decryption oracle BiDec by taking each query e of $\mathcal{A}$, and sending $\mathsf{Dec}_{k_1}(e)$ to its own decryption oracle. $\mathcal{A}$ chooses two messages (m_0, m_1) to challenge the bidirectional encryption scheme $\mathcal{E}'$ and sends them to $\mathcal{B}$. $\mathcal{B}$ uses the same two messages to challenge the standard encryption scheme $\mathcal{E}$. When $\mathcal{B}$ is presented with the challenge $\mathsf{Enc}_{k_2}(m_b)$, where $m_b \in (m_0, m_1)$, $\mathcal{B}$ sends to $\mathcal{A}$ $\mathsf{BiEnc}(m_b) = \mathsf{Enc}_{k_1}(\mathsf{Enc}_{k_2}(m_b))$. We assumed that P is able to break the bidirectional encryption scheme with non-begligible probability. Thus, $\mathcal{B}$ breaks the standard encryption scheme with non-negligible probability.

Generic	$\mathsf{Succ}_{P,\mathcal{S}} \overset{\text{def}}{=} \Pr\left[\mathsf{BiVer}(m,s) = succeed \;\middle|\; \begin{array}{l} (\mathsf{SK_U},\mathsf{VK_U},\mathsf{SK_F},\mathsf{VK_F},\pi) \leftarrow \mathsf{BiGen}(1^k) \\ (m,s) \leftarrow \mathsf{P}^{\mathsf{BiSig}}(\mathsf{VK_U},\mathsf{VK_F},\pi) \end{array} \right]$
	$\mathsf{Succ}_{F,\mathcal{S}} \overset{\text{def}}{=} \Pr\left[\mathsf{BiVer}(m,s) = succeed \;\middle|\; \begin{array}{l} (\mathsf{SK_U},\mathsf{VK_U},\mathsf{SK_F},\mathsf{VK_F},\pi) \leftarrow \mathsf{BiGen}(1^k) \\ (m,s) \leftarrow \mathsf{F}^{\mathsf{P}}(\mathsf{VK_U},\mathsf{VK_F},\mathsf{SK_F}) \end{array} \right]$

Table 9. Bidirectional signature definitions for adversary's success.

C.2 Bidirectional El Gamal Encryption Scheme

Theorem 8 *Let $\mathcal{E}' = (\mathsf{BiGen}, \mathsf{BiEnc}, \mathsf{BiDec}, \Pi)$ be an bidirectional El Gamal encryption scheme. $\mathcal{E}'$ is* CPA *secure against (1) the proxy* P, *(2) the user* F, *and (3) all users* U, *where the adversary's success is defined in table 8.*

Proof

1. Let's assume that P can break the bidirectional El Gamal encryption scheme. This means that $|\mathsf{Succ}_{P,\mathcal{E}}(1^k) - 1/2|$ is not negligible. Based on P, we will build an algorithm $\mathcal{B}$ that can break the standard El Gamal encryption. For this, $\mathcal{B}$ simulates the conditions required by P. Initially, $\mathcal{B}$ knows the public key of U ($\mathsf{EK_U} = g^{x_1}$). $\mathcal{B}$ pick a random number π and forwards it to P, together with $\mathsf{EK_U}$. From $\mathsf{EK_U} = g^{x_1}$ and π, $\mathcal{B}$ calculates $g^{x_2} = g^{x_1}g^{\pi}$ and forwards g^{x_2} to P as the secret key of F. P chooses two messages (m_0, m_1) to challenge the security of the bidirectional El Gamal and forwards them to $\mathcal{B}$. $\mathcal{B}$ considers the same two messages to challenge the standard El Gamal and receives the challenge $\mathsf{Enc}_{x_1}(m_b)$, where $m_b \in (m_0, m_1)$. $\mathcal{B}$ forwards the challenge $\mathsf{Enc}_{x_1}(m_b) = \mathsf{BiEnc}_{x_1}(m_b)$ to P. We considered that P can break the bidirectional El Gamal with probability of success $\mathsf{Succ}_{P,\mathcal{E}}(1^k)$ greater than 1/2. Thus, $\mathcal{B}$ is able to break the standard El Gamal with probability of success greater than 1/2.

2. Let's assume that F can break $\mathcal{E}'$. This means that $|\mathsf{Succ}_{F,\mathcal{E}}(1^k) - 1/2|$ is not negligible. The proof is similar to the previous one. The only difference is that $\mathcal{B}$ chooses at random x_2 and computes g^{x_2} for F.

3. Let's assume that $\mathcal{E}$ is not CPA secure. This means that there exists an adversary $\mathcal{A}$ such that $|\mathsf{Succ}_{\mathcal{A},\mathcal{E}}(1^k) - 1/2|$ is not negligible. The proof is similar to the previous one. The only difference is that $\mathcal{B}$ chooses at random $\mathsf{DK}_{\mathcal{A}} = y$ and computes $\mathsf{EK}_{\mathcal{A}} = g^y$ for $\mathcal{A}$.

D Bidirectional Signature Scheme

D.1 Bidirectional Generic Signature Scheme

Theorem 9 *Let's consider a standard signature scheme $\mathcal{S} = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$. Based on $\mathcal{S}$, we build an bidirectional signature scheme $\mathcal{S}' = (\mathsf{BiGen}, \mathsf{BiSig}, \mathsf{BiVer}, \Pi)$. If $\mathcal{S}$ is* UF *than $\mathcal{S}'$ is* UF *against (1) the proxy* P, *(2) the user* F, *and (3) all users* U, *where the adversary's success is defined in table 9.*

Proof

1. Let's assume that $\mathcal{S}'$ is not UF against P. This means that $|\mathsf{Succ}_{P,\mathcal{S}}(1^k)|$ is not negligible. Based on P, we build a forger $\mathcal{B}$ able to break the original signature $\mathcal{S}$. $\mathcal{B}$ tries to generate a valid signature $\mathsf{Sig}_1(m)$ for a message m. $\mathcal{B}$ receives VK_1 as input. $\mathcal{B}$ generates two random numbers $(\mathsf{SK}_2, \mathsf{SK}_3)$ and sends them to P as the bidirectional key π. The forger $\mathcal{B}$ starts P.

   When *proxy* makes a query for a message m to the hash oracle, $\mathcal{B}$ forwards the request to its own hash oracle and returns the answer $h(m)$ to P. When P makes a query to the π signature oracle, $\mathcal{B}$ makes a query to the Sig signature oracle for the same message m. $\mathcal{B}$ receives $\mathsf{Sig}_1(m)$, computes $\mathsf{Sig}_3(m)$ and sends to P $\mathsf{BiSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_3(m)$. At one moment, P generates a valid signature $\mathsf{BiSig}(m)$ for a new message m, with non-negligible probability. $\mathcal{B}$ takes $\mathsf{BiSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_3(m)$, ignores the second part and outputs $\mathsf{Sig}_1(m)$. We assumed that P can break the bidirectional signature scheme $\mathcal{S}'$. Thus, $\mathcal{B}$ can break the original signature scheme $\mathcal{S}$.

2. Let's assume that $\mathcal{S}'$ is not UF against F. This means that $|\mathsf{Succ}_{F,\mathcal{S}}(1^k)|$ is not negligible. Based on F, we build a forger $\mathcal{B}$ able to break the original signature $\mathcal{S}$. $\mathcal{B}$ tries to generate a valid signature $\mathsf{Sig}_2(m)$ for a message m. $\mathcal{B}$ receives VK_2 as input. $\mathcal{B}$ generates two pairs of random numbers $(\mathsf{VK}_1, \mathsf{SK}_1, \mathsf{VK}_3, \mathsf{SK}_3)$ and sends to F as its keys. The forger $\mathcal{B}$ starts F. When

fbi makes a query for a message m to the hash oracle, $\mathcal{B}$ forwards the request to its own hash oracle and returns the answer $h(m)$ to F. When F makes a query to the P signature oracle, $\mathcal{B}$ makes a query to the Sig signature oracle for the same message m. $\mathcal{B}$ receives $\mathsf{Sig}_2(m)$, computes $\mathsf{Sig}_1(m)$ and sends to F $\mathsf{BiSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_2(m)$. At one moment, F generates a valid signature $\mathsf{BiSig}(m)$ for a new message m, with non-negligible probability. $\mathcal{B}$ takes $\mathsf{BiSig}(m) = \mathsf{Sig}_1(m)\mathsf{Sig}_2(m)$, ignores the first part and outputs $\mathsf{Sig}_2(m)$. We assumed that F can break the bidirectional signature scheme $\mathcal{S}'$. Thus, $\mathcal{B}$ can break the original signature scheme $\mathcal{S}$.

3. This part is implied by the two previous parts.