D. Nuñez, I. Agudo, and J. Lopez, "Escrowed decryption protocols for lawful interception of encrypted data", *IET Information Security*, vol. 13, pp. 498 – 507, 2019.
NICS Lab. Publications: `https://www.nics.uma.es/publications`

Escrowed Decryption Protocols for Lawful Interception of Encrypted Data

David Nuñez, Isaac Agudo, and Javier Lopez

Network, Information and Computer Security Laboratory (NICS Lab)

Universidad de Málaga, Spain

{*dnunez,isaac,jlm*}*@lcc.uma.es*

Abstract

Escrowed decryption schemes are public-key encryption schemes with an *escrowed decryption* functionality that allows authorities to decrypt encrypted messages under investigation, following a protocol that involves a set of trusted entities called *custodians*; only if custodians collaborate, the requesting authority is capable of decrypting encrypted data. This type of cryptosystems represents an interesting trade-off to the privacy vs. surveillance dichotomy. In this paper, we propose two escrowed decryption schemes where we use proxy re-encryption to build the escrowed decryption capability, so that custodians re-encrypt ciphertexts, in a distributed way, upon request from an escrow authority, and the re-encrypted ciphertexts can be opened only by the escrow authority. Our first scheme, called EDS, follows an all-or-nothing approach, which means that escrow decryption only works when all custodians collaborate. Our second scheme, called TEDS, supports a threshold number of custodians for the escrow decryption operation. We propose definitions of semantic security with respect to the authorities, custodians and external entities, and prove the security of our schemes, under standard pairing-based hardness assumptions. Finally, we present theoretical and experimental analysis of performance of both schemes, which shows that they are applicable to real-world scenarios.

1 Introduction

There is a growing debate nowadays around the dichotomy between data confidentiality and law enforcement investigations in digital communication networks, such as the Internet. On one side of this discussion, governments and law enforcement agencies (LEAs) are concerned with the alleged impunity that is derived from the use of private and confidential communications. This has motivated the proposal of surveillance mechanisms as a means to detect and prevent illegal activities (e.g., child pornography) and national security threats (e.g., terrorism). There is no doubt that the abstract goal of protecting citizens from these dangers is noble; the same cannot be said of all the actual methods to achieve this. Contrary to this view, a large part of the society, which includes private citizens and corporations, see these claims as a threat to their rights to privacy and confidentiality. Relatively recent examples of this are the controversy produced by the US' Investigatory Powers Act 2016 [1] and the dispute betwee FBI and Apple [2]. There is an intense, on-going debate nowadays regarding whether mechanisms for breaking confidentiality of communications are a legitimate method to fight the mentioned threats.

One of the most immediate concerns is the perceived lack of accountability from the government and LEAs; in other words, if such mechanisms were available, there is no hindrance for the government and LEAs to use it as an effective means for mass surveillance. According to a document exposed by the Washington Post [3], a working group of the US government delivered an internal document containing a list of principles with respect to the use of encryption, and its relation with industry and law enforcement. Of particular interest are the following principles: "*2. No unilateral government access. Approaches should not provide "golden keys" to government or allow government to access information without the assistance of a third party. 3. Technologically-enforced limits. To the extent possible, approaches should rely on technology, rather than procedural protections, to enforce constraints on government access*".

The inclusion of independent third parties as necessary parts of "government access", enforced by technological means, seems like a plausible trade-off in the dichotomy between data confidentiality and

law enforcement investigations. Aligned with the aforementioned principles, [4] proposed a protocol for *Accountable Escrowed Encryption*, which, on the one hand, allows escrow authorities (a term that subsumes government and LEAs) to decrypt encrypted messages under investigation, but on the other hand, enables to hold them accountable. Their proposal consists on a special encryption scheme that has a built-in *escrowed decryption* capability. In order to use this capability, escrow authorities have to follow a protocol involving a set of trusted entities called *custodians*; only if all custodians collaborate, the requesting escrow authority is capable of decrypting the encrypted message. A core principle of this solution is that it does not require *key escrow* (i.e., a "backdoor" in the encryption scheme for extracting users' private key); instead, only the decryption capability is escrowed, on a case-by-case basis. Another interesting aspect of this proposal is that the custodians are in a position of providing accountability by informing citizens of the number of escrow requests; this is implemented by recording such information on a public log, facilitating the society to react through democratic procedures (e.g., demanding "less/more decryption" from escrow authorities). The authors argue that, this way, a balance between societal security and individual privacy can be achieved.

In this paper we propose an alternative construction for the Accountable Escrowed Encryption scheme by [4]. Our proposal solves several problems from the original scheme, related to security and efficiency. The proposed construction is similar in essence but borrows techniques from Proxy Re-Encryption [5] in order to build the escrowed decryption capability. That is, the trusted custodians re-encrypt ciphertexts, in a distributed way, upon request from the escrow authority, and the re-encrypted ciphertexts can be opened by the escrow authority.

We note that lawful interception is not the only suitable application of the proposed systems. For example, these systems can also be used to implement data recovery in corporate scenarios, where data is encrypted with public keys belonging to users of the system, and under special circumstances, it can be delegated to authorized escrow agents (e.g., internal auditor, system administrator, etc) in a controlled manner. However, lawful interception scenarios seem much more illustrative and their demand is more pressing, and therefore, in the rest of the article we will focus on this use case.

1.1 Contributions

In this paper we make the following contributions:

- Motivated by the need of a trade-off between confidentiality and lawful interception, we present a high-level definition of escrowed decryption schemes, building upon the work of [4], and provide a comprehensive description of actors, interactions, syntax and trust assumptions for this type of schemes.
- We define two security notions for realizing semantic security in escrowed decryption schemes: one with respect to the escrow authority (IND-EA-CPA), and another with respect to the custodians (IND-CUST-CPA).
- We propose the Escrowed Decryption Scheme (EDS) cryptosystem, which uses proxy re-encryption techniques to instantiate the escrowed decryption capability. We show that EDS is secure under the previous notions assuming the hardness of the SXDH problem.
- We propose a threshold variant of EDS, called Threshold EDS (or TEDS), which is also secure under the previous notions assuming the hardness of the SXDH problem.
- We analyze the performance of EDS and TEDS, both from the theoretical and experimental points of view.

1.2 Organization

The rest of this paper is organized as follows: In Section 2, we present related state of the art, and in Section 3, we briefly summarize some cryptographic concepts that will be used later. In Section 4, we describe in detail the escrowed decryption system, including actors and their interaction, trust assumptions, design goals, and other formalizations. In Section 5, we present a first proposal of escrowed decryption system, named EDS, including an analysis of correctness and security, and in Section 6, we propose a threshold variation, called TEDS. In Section 7, we analyze the computational costs of our proposals and describe the experimental results from a prototype implementation. Finally, Section 8 concludes the paper and future work is outlined.

2 Related Work

As described in the introduction, our proposal is inspired by the work of [4], and intended to solve some of its problems. One of them is that a collusion of custodians can potentially decrypt any message if they have access to a full ciphertext. That is, one have to make the assumption that the custodians are trusted for not doing this. In our proposal, the custodians are never able to see the underlying message, even if they have access to the whole ciphertext. Another problem of the original scheme is the efficiency of the solution. The protocol for escrow decryption in the original proposal is composed of 2 synchronous rounds, and each round involves all custodians; that is, the escrow authority has to first engage all custodians in a first round of interactions (ideally in parallel), and only when all of them have responded, it can continue with a second round of interactions. The reason behind this characteristic is that the escrow authority has to make some intermediate computations that are required for the second round, and it needs all the responses from the first round to do this. In our proposal, we reduce the escrow decryption protocol to a single round.

There are other related proposals in the literature, starting from the seminal works of [6], [7], and [8], all of them oriented towards key escrow (i.e., recovery the secret key of a user). [9] describe protocols to enhance accountability in case of mandated law enforcement access to stored data. However, one of the initial premises of these protocols is that the original data source stores the target information in the clear (e.g., a telecommunications carrier storing metadata about incoming/outgoing phone calls). In addition, users are left out from their proposal. [10] present a group signature scheme that allows to reveal the identity of a sender to authorities, in a covert, yet accountable way. Note that this work is focused on signatures and privacy of the sender, rather than encryption and confidentiality of messages.

Regarding the intersection of proxy re-encryption and distributed/threshold re-encryption, we should note the work of [11] in the context of certificateless proxy re-encryption, where they define a distributed re-encryption process based on Shamir's secret sharing. There are also other related proposals, such as [12, 13, 14], but in all of them the decryption key is distributed among several entities, rather than the re-encryption capability, which makes them closer to the key escrow literature, rather than to the concept of escrowed decryption.

3 Preliminaries

This section introduces some cryptographic basic concepts that will be used during the paper. First, we introduce bilinear pairings, which are used in our proposed schemes, as well as some associated hardness assumptions. Next, we briefly describe the concepts of proxy re-encryption and secret sharing, since we use ideas from both types of cryptosystems.

3.1 Bilinear pairings and hardness assumptions

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order q. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ satisfying the following properties:

- Bilinearity: For all $a, b \in \mathbb{Z}_q$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, it holds that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Non-degeneracy: For all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, it holds that $e(g_1, g_2) \neq 1$.
- Computability: There is an efficient algorithm that computes e.

Depending on the characteristics of the groups, there are essentially three types of pairings [15]:

- Type-1 pairings: $\mathbb{G}_1 = \mathbb{G}_2$.
- Type-2 pairings: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficiently computable homomorphism $\phi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- Type-3 pairings: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism between $\mathbb{G}_2$ and $\mathbb{G}_1$.

Type-1 pairings are said *symmetric* (since $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same group), and Type-2 and Type-3 pairings are *asymmetric*. It is advisable to design cryptographic schemes and protocols using Type-3 pairings, since they achieve the best trade-off between security and efficiency [15, 16], although there have been some recent proposals that allow to design protocols in a Type-1 setting and to automatically translate them to Type-3 [17, 18]. For simplicity in the design process, we directly use Type-3 pairings. The following is a common hardness assumption for this type of pairings:

Definition 1 (The External Diffie-Hellman assumption (XDH) [19]). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be pairing groups of order q. Given a tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1^4$, where $a, b, c \in \mathbb{Z}_q$, the XDH assumption holds if it is not computationally feasible to decide whether $g^c = g^{ab}$. In other words, the DDH problem in $\mathbb{G}_1$ is hard.*

When, in addition to assuming that the DDH is hard in $\mathbb{G}_1$, we consider the DDH hard in $\mathbb{G}_2$, then it is called the Symmetric External Diffie-Hellman assumption.

Definition 2 (The Symmetric External Diffie-Hellman assumption (XDH) [20]). *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be pairing groups. The SXDH assumption holds if the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

3.2 Proxy Re-Encryption

From a high-level viewpoint, proxy re-encryption is a type of public-key encryption that enables a proxy to transform ciphertexts under Alice's public key into ciphertexts decryptable by Bob's secret key. This transformation is performed by a proxy entity using a re-encryption key generated by Alice, without revealing to the proxy any information about the underlying message. There are multiple proxy re-encryption proposals in the literature (see [21] for an overview), being the most prominent the ones by [22] and [5].

3.3 Shamir's Secret Sharing

Let us assume that a dealer wants to share a secret $s \in \mathbb{Z}_q$ among n participants, so that it can be reconstructed with t "*shares*". The dealer first samples random coefficients $f_1, f_2, ..., f_{t-1} \in \mathbb{Z}_q$ and defines a polynomial $f(x) \in \mathbb{Z}_q[x]$, so that:

$$f(x) = s + f_1 x + f_2 x^2 + ... + f_{t-1} x^{t-1}$$

Now, for each participant $i \in \{1, ..., N\}$, the dealer sends the share $f(i)$. It can be seen that the secret shared by the dealer can be obtained as $f(0)$. Using Lagrange's polynomial interpolation, the secret can be reconstructed from any set of t shares. Let $\mathcal{I} \subseteq \{1, ..., N\}$ be a set of t indices. Reconstruction of the shared secret can be done as follows:

$$s = \sum_{i \in \mathcal{I}} f(i) \cdot \lambda_i, \quad \text{where } \lambda_i = \prod_{j \in \mathcal{I}/\{i\}} \frac{j}{j - i} \tag{1}$$

This cryptosystem is due to [23] and provides information-theoretic security of the secret (that is, with less than t shares, all possible values in $\mathbb{Z}_q$ are equally probable, so an adversary has no information in this case).

4 System model

This section is devoted to the definition of what constitutes an escrowed decryption scheme. First, we present a general overview of such schemes, their actors and interactions, as well as the corresponding trust assumptions. Second, we describe their design goals. Third, we provide a formal definition and syntax for our proposed cryptosystems. Finally, we propose security models that cover some of the previous design goals from a cryptographic point of view.

4.1 Overview

The ultimate goal of an escrowed decryption scheme is to provide technical means to a legitimate entity to decipher encrypted messages, but only with the collaboration of other independent, trusted third-parties that can limit the process. Similarly to [4], we consider four main types of actors in an escrowed decryption system, which are the following:

- Users, which are entities that want public and private keys to engage in encrypted communications with others.
- Certification Authority (CA), which is in charge of providing certified public keys to users, and ensuring that the escrowed decryption capabilities are guaranteed. These capabilities are embodied by a set of *escrow shares*, which are cryptographic material intended for the custodians that enables them to assist in escrow decryption.

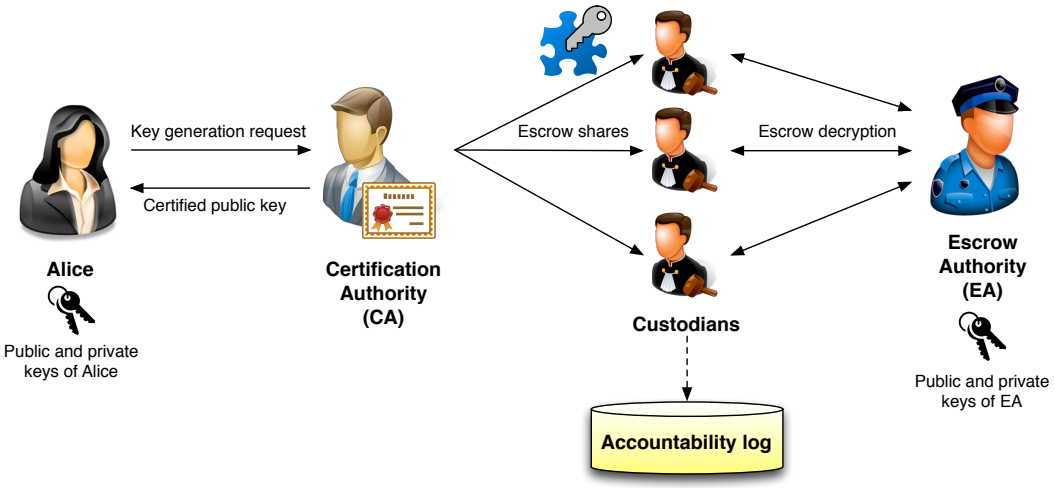

Figure 1: High-level view of actors and interactions in an escrowed decryption system

- Custodians are trusted entities that participate in an escrowed decryption procedure, using an escrow share. Custodians may (or may not) positively respond to access requests from escrow authorities on a case-by-case basis, depending on legal or technical motives. Optionally, custodians may record in a public accountability log any escrow decryption request received from escrow authorities.
- Escrow Authority (EA), which is an entity that wants to access to information that is encrypted and that interacts with a set of custodians to perform an escrowed decryption of such information. We assume that the EA has a pair of public and private keys. For simplicity, we will assume only one escrow authority in the system. Examples of escrow authorities are governments or Law Enforcement Agencies (LEAs) in criminal investigation scenarios, or internal auditors in a corporate environment.

Figure 1 depicts a high-level view of the actors that participate in an escrowed decryption system, as well as their interactions.

4.2 Trust assumptions

The protocol has the following trust assumptions:

- The Escrow Authority is assumed to eavesdrop and intercept any communication between users of the system.
- The custodians are assumed to be honest-but-curious, which means they behave correctly, in accordance to the expectations, but at the same time, they are interested in learning the underlying information.
- The Certification Authority is also honest-but-curious. In our proposed schemes, it can potentially replace the role of the custodians, and for this reasons, we require it to be an honest actor. However, it may attempt to gain knowledge about the messages and should not be able to do so.
- Collusions of Escrow Authority and Custodians are not considered, since they trivially allow to decrypt communications. In fact, a legitimate escrowed decryption procedure is basically a collaboration between the escrow authority and the custodians, which makes collusions of escrow authorities and custodians no different from regular, legitimate interactions. Since the CA can potentially replace the custodians (as discussed in the previous item), collusions of escrow authority and CA are also not permitted.

4.3 Design goals

We set the following design goals, which will be used later for the security analysis:

1. No key escrow: The system should not include a "backdoor" that allows extracting users' private key. Instead, only the decryption capability is escrowed, on a case-by-case basis.

2. Escrow authorities must not be able to learn any information from an encrypted message without the participation of the required custodians.

3. Custodians must not learn any underlying information from the encrypted messages during the escrowed decryption process.

4. It should not be possible to impersonate an escrow authority. In other words, only legitimate escrow authorities can successfully decrypt intercepted messages, with collaboration of the custodians.

5. Custodians are in a position of providing accountability by publicly informing of the identity of affected users and/or number of escrow decryption requests. We consider this goal as optional, since it may not be cryptographically enforced by the encryption scheme.

4.4 Formal definition

The idea behind our proposed schemes is to base the escrowed decryption protocol on a "shared" re-encryption process by the custodians, in a way reminiscent to the decryption procedure of a threshold encryption scheme [24]; that is, our solution ensures that the escrow authority is able to decrypt ciphertexts intended for suspicious users as long as he engages the collaboration of all the escrow custodians.

Formally, we define a new type of cryptosystem that consists of a public-key encryption scheme with an added escrowed decryption capability, which we construct using proxy re-encryption techniques. In order to use this capability, escrow authorities have to follow a protocol involving a set of custodians, and only if all of them collaborate, the requesting escrow authority is capable of decrypting the encrypted message. The following is the generic syntax of this new cryptosystem:

- $\mathsf{Setup}(1^k) \rightarrow pp$. On input the security parameter 1^k, the setup algorithm outputs the set of global parameters pp, which includes the public and private key of the escrow authority (pk_{EA}, sk_{EA}).
- $\mathsf{KeyGen}(pk_{EA}) \rightarrow (pk, sk, \{\kappa_i\}_{i=1}^n)$. On input the escrow authority's public key pk_{EA}, the key generation algorithm outputs the public and private key of user U, (pk, sk), and a set of *escrow shares*, $\{\kappa_i\}_{i=1}^n$.
- $\mathsf{Enc}(pk, m) \rightarrow CT$. On input a public key pk and a message m, the encryption algorithm outputs ciphertext CT.
- $\mathsf{Dec}(sk, CT) \rightarrow m$. On input the secret key sk and a ciphertext CT, the decryption algorithm outputs the original message m.
- $\mathsf{ShareReEnc}(\kappa_i, CT) \rightarrow \rho_i$. On input an escrow share κ_i and a ciphertext CT, this algorithm outputs the *re-encryption share* ρ_i.
- $\mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i=1}^n) \rightarrow m$. On input the escrow authority's secret key sk_{EA}, a ciphertext CT, and a set of re-encryption shares $\{\rho_i\}_{i=1}^n$, the combination algorithm outputs the original message m.

4.5 Security model

We consider security against three types of adversaries, namely the escrow authority, the custodians, and external entities:

4.5.1 Security against the escrow authority

As usual in most encryption schemes, it is necessary to provide a formal assurance of security, at least based on an indistinguishability notion. Here we target indistinguishability against chosen-plaintext attacks (IND-CPA), although adapted to our proposed cryptosystem, in particular with respect to the escrow authority. Apart from the usual characteristics of the IND-CPA game in public-key encryption, we also allow the adversary to know the secret key of the escrow authority (EA), which is equivalent to consider the escrow authority as the adversary. Informally, this means that the EA should not be able to distinguish messages based on their encryption. We named this security notion as IND-EA-CPA, and

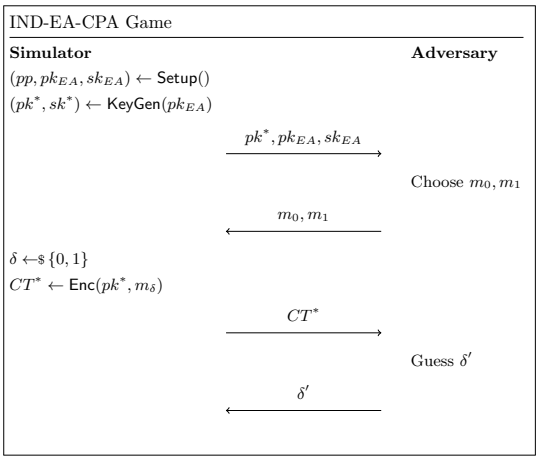

Figure 2: Description of the IND-EA-CPA Game

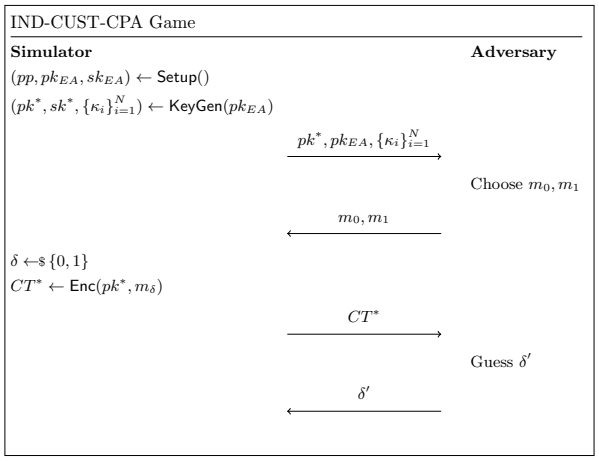

Figure 3: Description of the IND-CUST-CPA Game

the associated game is represented in Figure 2. The basic operation of this game is the following: given a user's public key, an adversary chooses two messages and asks for a challenge ciphertext, which is the encryption of one of these messages under the user's public key; satisfaction of this security notion implies that the adversary should not be able to distinguish which message is encrypted (or more accurately, his advantage with respect to random guessing should be negligible), even if he knows the secret key of the escrow authority. It can be seen that this security notion covers the second design goal (see Section 4.3).

Although we use techniques from proxy re-encryption in our proposed cryptosystems, we stress that we are not targeting here a security notion suitable for PRE, since this is not a PRE scheme. In particular, it is only possible to "re-encrypt" to the escrow authority, not to any other regular user. For the same reason, and given that in this security model the adversary knows the secret key of the escrow authority, we cannot give him also the escrow shares (which are equivalent to re-encryption keys, in PRE terms), since this would make the game trivial (i.e., that is, the adversary could use the escrow shares to re-encrypt ciphertexts, and the secret key of the escrow authority to decrypt it). We also note that, since we are targeting a CPA model, there is no need to describe any oracle.

4.5.2 Security against the custodians

The other security notion we target is IND-CPA with respect to the custodians, which can be seen as symmetric to the first one. In particular, we allow the adversary to know all the escrow shares, which is equivalent to consider the custodians as the adversary. We named this security notion as IND-CUST-CPA, and the associated game is represented in Figure 3. Analogously to the previous game, we cannot provide the adversary with the secret key of the escrow authority, since this would make the game trivial. This security notion covers the third design goal (see Section 4.3).

4.5.3 Security against external adversaries

We consider that this adversary has no additional knowledge apart from public information. Therefore, in this case, we are dealing with the traditional IND-CPA security notion. It can be shown that it is strictly weaker than the previous notions (that is, satisfaction of any of them implies satisfaction of this notion, since the adversary has less information in the latter case). For this reason, we will not consider it explicitly in our security analysis. This security notion covers the fourth design goal (see Section 4.3).

5 Escrowed Decryption Scheme (EDS)

In this section we formally describe our proposed Escrowed Decryption Scheme (EDS), as well as prove its correctness and analyze its security under the models described before.

From the point of the of the design of the cryptographic scheme, we base our solution on a proxy re-encryption scheme (in particular, the one by [5]). Roughly, we implement the escrowed decryption process as a re-encryption of a ciphertext, so that it can be decrypted by the escrow authority. There are, however, two main modifications with respect the scheme of Ateniese et al.:

- Re-encryption is split among several entities (i.e., the custodians). The re-encryption key is split into N shares, so $rk = \prod_{i=1}^{N} rk_i$. Then, for the decryption of re-encrypted ciphertexts, we make use of the multiplicative homomorphic properties of the scheme. Informally:

$$\mathsf{ReEnc}(rk, CT) = \prod_{i=1}^{N} \mathsf{ReEnc}(rk_i, CT)$$

  In our scheme, these shares are called *escrow shares* and represented by κ_i.

- The original scheme from Ateniese et al. allows to encrypt ciphertexts that are not re-encryptable (called "first-level ciphertexts"), which can be used to bypass our re-encryption-based escrow. We get rid of this functionality by modifying the key generation in such a way that users' public key cannot be used to create first-level ciphertexts.

5.1 Description of the scheme

Our scheme is defined as follows. Note that although in the generic syntax above we described functions, our solution requires some of them to be implemented as two-party protocols. In particular, key generation is jointly performed by the user and the CA, while escrow decryption requires individual interaction of the escrow authority with all the custodians.

Setup. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a type-3 pairing, g a generator of $\mathbb{G}_1$, h a generator of $\mathbb{G}_2$, with $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order q, and $Z = e(g, h)$. The public key of the escrow authority is $pk_{EA} = h^a$, and the corresponding secret key $sk_{EA} = a$, with $a \in \mathbb{Z}_q$ sampled uniformly at random. This public key is certified by the CA in the traditional way. Let $N > 0$ be the number of custodians. The public parameters of the system are the elements of the tuple $pp = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, Z, pk_{EA}, N)$

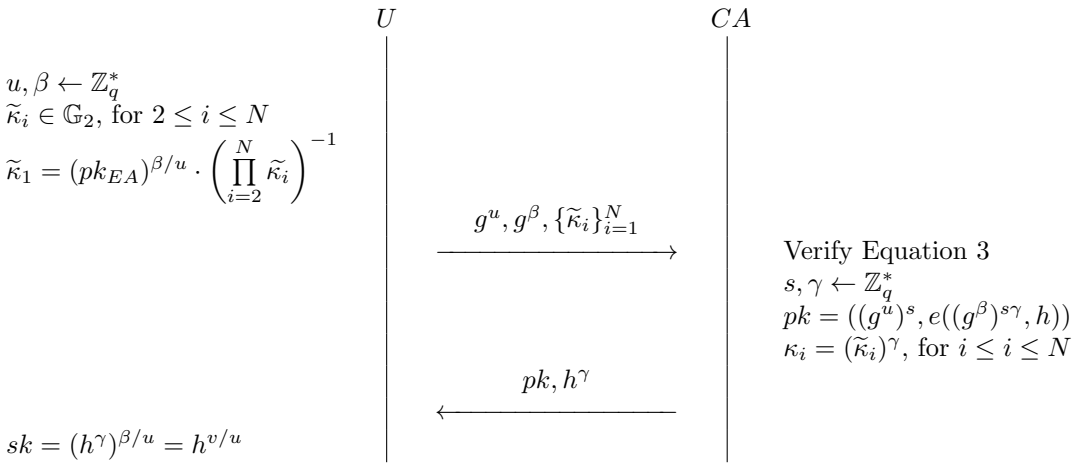

Figure 4: Key Generation subprotocol

Key Generation. The user U first selects random secrets $u, \beta \in \mathbb{Z}_q$, and computes g^u and g^β. Let $\mathcal{C}$ be the set of indices of the custodians; for simplicity, we can assume that $\mathcal{C} = \{1, ..., N\}$. Now, he chooses random $\widetilde{\kappa}_i \in \mathbb{G}_2$, for $2 \leq i \leq N$, and computes $\widetilde{\kappa}_1$ as follows:

$$\widetilde{\kappa}_1 = (pk_{EA})^{\beta/u} \cdot \left(\prod_{i=2}^{N} \widetilde{\kappa}_i \right)^{-1}$$

These elements $\widetilde{\kappa}_i$ are *partial escrow shares*, and it can be seen that they satisfy the relation:

$$\prod_{i=1}^{N} \widetilde{\kappa}_i = h^{a\beta/u} \tag{2}$$

Next, he sends $(g^u, g^\beta, \{\widetilde{\kappa}_i\}_{i=1}^N)$ to CA for certification. CA verifies that the input received is valid by checking that the following equation holds:

$$e(g^u, \prod_{i=1}^{N} \widetilde{\kappa}_i) = e(g^\beta, pk_{EA}) \tag{3}$$

Next, CA chooses random $s, \gamma \in \mathbb{Z}_q$ and computes the public key of user U as follows:

$$pk = ((g^u)^s, e((g^\beta)^{s\gamma}, h)) = (g^{su}, Z^{sv})$$

It can be seen that this implicitly defines $v = \beta \cdot \gamma$, without the user knowing γ and the CA knowing β. This public key is then signed and certified by the CA, in the traditional way of public-key infrastructure. Now, the CA concludes the generation of the set of escrow shares, by computing $\kappa_i = (\widetilde{\kappa}_i)^\gamma$ for each custodian. Note that the following relation between escrow shares and the escrow authority's public key holds:

$$\prod_{i=1}^{N} \kappa_i = (\prod_{i=1}^{N} \widetilde{\kappa}_i)^\gamma = (h^{a\beta/u})^\gamma = h^{av/u} = (pk_{EA})^{v/u} \tag{4}$$

The CA sends each escrow share to each of the N custodians through a secure channel. Finally, CA returns (pk, h^γ) to user U, who sets $sk = (h^\gamma)^{\beta/u} = h^{v/u}$. Figure 4 shows the interactions of this subprotocol.

Encryption. Any entity can encrypt message $m \in \mathbb{G}_T$ into a ciphertext under user U's public key $pk = (g^{su}, Z^{sv})$ by choosing a random $r \in \mathbb{Z}_q$ and outputting the tuple:

$$CT = ((g^{su})^r, (Z^{sv})^r \cdot m) = (g^{sur}, Z^{svr} \cdot m)$$

Decryption. User U can decrypt a ciphertext $CT = (CT_1, CT_2) = (g^{sur}, Z^{svr} \cdot m)$ using his secret key $sk = h^{v/u}$ as follows:

$$m = \frac{CT_2}{e(CT_1, sk)}$$

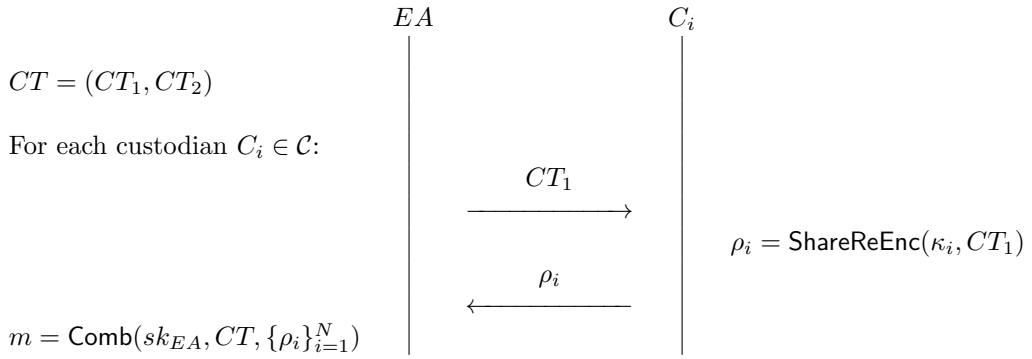

Figure 5: Escrowed Decryption subprotocol

Escrowed decryption. This subprotocol is basically the combination of the $\mathsf{ShareReEnc}$ and Comb algorithms, as seen in Figure 5, where the $\mathsf{ShareReEnc}$ algorithm is performed by the custodians and the Comb algorithm is done by the escrow authority. When the escrow authority wants to decrypt a ciphertext $CT = (CT_1, CT_2) = (g^{sur}, Z^{svr} \cdot m)$ that is encrypted under user U's public key, they give $CT_1 = g^{sur}$ to each custodian, and obtain as response the re-encryption share ρ_i, computed as follows:

$$\rho_i = \mathsf{ShareReEnc}(CT_1, \kappa_i) = e(g^{sur}, \kappa_i)$$

Note that although we defined the syntax of $\mathsf{ShareReEnc}$ such that it takes the whole ciphertext as input, in our proposed scheme only the first component is necessary, which allows to save some bandwidth. After getting the re-encryption shares from all custodians, denoted by the set $\{\rho_i\}_{i=1}^N$, the escrow authority executes the combination algorithm Comb to decrypt the ciphertext and obtain the original message m. In our proposal, this algorithm is defined as follows:

$$\mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i=1}^N) = \frac{CT_2}{(\prod_{i=1}^{N} \rho_i)^{1/sk_{EA}}}$$

5.2 Correctness

The correctness of our proposal is demonstrated by showing the consistency of the key generation, decryption, and escrowed decryption procedures:

Key Generation The goal of this procedure is to produce correct public keys and escrow shares, through a protocol between the user and the CA. Correctness is ensured by Equation 3, which checks that the relation between the inputs provided by the user (i.e., $(g^u, g^\beta, \{\widetilde{\kappa}_i\}_{i=1}^N)$) and the public key of escrow authority (pk_{EA}) is valid. The value of g^u and g^β is inherently arbitrary (since the user chooses both u and β at random), so the only place where an inconsistency may occur is in the $\widetilde{\kappa}_i$ elements. It can be seen that the only way to pass the CA validity check (Equation 3) is that Equation 2 holds. As long as the user provides inputs following this latter relation, the correctness of key generation is ensured; otherwise, it is detected by the CA with the validity check.

More formally, it can be seen that Equation 3 holds when the escrow shares are consistent with g^u, g^β and pk_{EA}:

$$\begin{aligned} e(g^u, \prod_{i=1}^{N} \widetilde{\kappa}_i) &= e(g^u, h^{a\beta/u}) \\ &= e(g, h)^{a\beta} \\ &= e(g^\beta, h^a) = e(g^\beta, pk_{EA}) \end{aligned}$$

Decryption Let $CT = (CT_1, CT_2) = (g^{sur}, Z^{svr} \cdot m)$ be a ciphertext encrypted under user U's public key. It can be seen that the regular decryption procedure works since:

$$\mathsf{Dec}(sk, CT) = \frac{CT_2}{e(CT_1, sk)} = \frac{Z^{svr} \cdot m}{e(g^{sur}, h^{v/u})} = m$$

Escrowed Decryption Assuming the Key Generation subprotocol is correct (which is ensured by the CA), then the escrowed decryption procedure is also correct, as long as the custodians respond correctly to the queries from the escrow authority (which are assumed honest-but-curious in our system). Recall that when the escrow authority wants to decrypt a ciphertext CT, he gives CT_1 to each custodian in $\mathcal{C}$, and obtains the re-encryption share $\rho_i = \mathsf{ShareReEnc}(CT_1, \kappa_i) = e(g^{sur}, \kappa_i)$ in response. In the next step, the escrow authority combines all the re-encryption shares ρ_i, so that:

$$\prod_{i=1}^{N} \rho_i = \prod_{i=1}^{N} e(g^{sur}, \kappa_i)$$

Thus, by the bilinear property of the pairing:

$$\prod_{i=1}^{N} \rho_i = e(g^{sur}, \prod_{i=1}^{N} \kappa_i)$$

Next, since the Key Generation subprotocol ensures that $\prod_{i=1}^{N} \kappa_i = h^{av/u}$ (see Equation 4), then:

$$\prod_{i=1}^{N} \rho_i = e(g^{sur}, h^{av/u}) = Z^{savr}$$

Finally, in the combination algorithm, the escrow authority uses this value to decrypt the ciphertext:

$$\mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i=1}^N) = \frac{Z^{svr} \cdot m}{(Z^{savr})^{1/a}} = m$$

5.3 Security analysis

In order to evaluate to security of our proposal, we prove that it complies with the notions described in Section 4.5:

5.3.1 Security against the escrow authority

Theorem 1 (EDS is IND-EA-CPA-secure)**.** *If the DDH problem in $\mathbb{G}_1$ is hard, then the EDS cryptosystem is secure under the IND-EA-CPA notion.*

Proof. The strategy in this proof is to embed an instance of the $\mathbb{G}_1$-DDH tuple in the IND-EA-CPA game (in particular, into the user's public key and the challenge ciphertext), which constitutes a reduction from the DDH problem in $\mathbb{G}_1$ to the IND-EA-CPA security of our cryptosystem.

Let us assume that there is an adversary $\mathcal{B}$ that wins the IND-EA-CPA game with non-negligible advantage ε. Then, we can use $\mathcal{B}$ to construct an algorithm $\mathcal{A}$ that solves the DDH problem in $\mathbb{G}_1$ with the non-negligible advantage. $\mathcal{A}$ receives a DDH tuple $(g, g^u, g^r, g^d) \in \mathbb{G}_1^4$, and his goal is to decide whether $d = u \cdot r$; he uses this tuple to simulate the environment for the adversary $\mathcal{B}$.

First, $\mathcal{A}$ simulates the output of the setup. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a type-3 pairing, g a generator of $\mathbb{G}_1$, h a generator of $\mathbb{G}_2$, with $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order q, and $Z = e(g, h)$. $\mathcal{A}$ samples random $a \in \mathbb{Z}_q$, sets the escrow authority's public and private keys $(pk_{EA}, sk_{EA}) = (h^a, a)$, and publishes the global parameters $pp = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, Z, pk_{EA}, N)$, as usual.

The next step is simulating the output of key generation subprotocol for the target user, that is, producing pk^*. $\mathcal{A}$ samples random $s, \beta, \gamma \in \mathbb{Z}_q$, and sets the public key of the target user as $pk^* = ((g^u)^s, Z^{s\beta\gamma}) = (g^{su}, Z^{sv})$, which is a valid public key, but with the value g^u from the DDH tuple embeded. Now $\mathcal{A}$ gives $\mathcal{B}$ the tuple (pk^*, pk_{EA}, sk_{EA}), as defined in the IND-EA-CPA game. Note that $\mathcal{A}$ does not release to $\mathcal{B}$ the user's secret values and the escrow shares, since any of these keys would allow to corrupt the target user and would make $\mathcal{B}$ win the game trivially.

Once the adversary provides the messages m_0 and m_1, $\mathcal{A}$ takes a random $\delta \in \{0, 1\}$, and constructs the challenge ciphertext using the values g^d and g^r from the DDH tuple as follows:

$$CT^* = ((g^d)^s, m_\delta \cdot e(g^r, h)^{s\beta\gamma}) = (g^{sd}, m_\delta \cdot Z^{svr})$$

$\mathcal{A}$ uses adversary $\mathcal{B}$ to obtain the guess δ' and decides that $d = u \cdot r$ when $\delta = \delta'$. Note that when $d = u \cdot r$, the challenge ciphertext is a valid encryption of m_δ under pk^*:

$$CT^* = ((g^{us})^r, m_\delta \cdot (Z^{sv})^r)$$

Therefore, in this case $\mathcal{A}$ solves the DDH problem in $\mathbb{G}_1$ with non-negligible advantage ε, since by initial assumption, $\mathcal{B}$ guesses δ correctly with the same advantage. On the contrary, when d is random, m_δ is information-theoretically hidden, so the advantage of $\mathcal{B}$ is 0. Thus, the overall success probability of $\mathcal{A}$ is $\frac{1}{2} + \frac{\varepsilon}{2}$, and therefore, $\mathcal{A}$ wins the IND-EA-CPA game with non-negligible advantage $\frac{\varepsilon}{2}$. □

5.3.2 Security against the custodians

Theorem 2 (EDS is IND-CUST-CPA-secure)**.** *If the DDH problem in $\mathbb{G}_2$ is hard, then the EDS cryptosystem is secure under the IND-CUST-CPA notion.*

Proof. The strategy in this proof is similar than in the previous case, but using a DDH tuple in $\mathbb{G}_2$, and embedding it into the target user's public key and the escrow shares. Let us assume that there is an adversary $\mathcal{B}$ that wins the IND-CUST-CPA game with non-negligible advantage ε. Then, we can use $\mathcal{B}$ to construct an algorithm $\mathcal{A}$ that solves the DDH problem in $\mathbb{G}_2$ with non-negligible advantage.

$\mathcal{A}$ receives a DDH tuple $(h, h^a, h^v, h^d) \in \mathbb{G}_2^4$, and his goal is to decide whether $d = a \cdot v$; he uses this tuple to simulate the environment for the adversary $\mathcal{B}$.

First, $\mathcal{A}$ simulates the output of the setup. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a type-3 pairing, g a generator of $\mathbb{G}_1$, h a generator of $\mathbb{G}_2$, with $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order q, and $Z = e(g, h)$. $\mathcal{A}$ sets the escrow authority's public key as $pk_{EA} = h^a$, based on the input DDH tuple, and publishes the global parameters $pp = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g, h, Z, pk_{EA}, N)$, as usual. Hence, $\mathcal{A}$ does not know the secret key of the escrow authority.

The next step is simulating the output of the key generation subprotocol; in this case, $\mathcal{A}$ releases to $\mathcal{B}$ the public key of the target user pk^* and the escrow shares κ_i of all the custodians. In order to do this, $\mathcal{A}$ samples random $s, u \in \mathbb{Z}_q$, and sets the public key of the target as $pk^* = (g^{su}, e(g, h^v)^s) = (g^{su}, Z^{sv})$, taking h^v from the DDH tuple. As for the escrow shares, he chooses random $\kappa_i \in \mathbb{G}_2$, for $2 \leq i \leq N$, and, taking h^d from the DDH tuple, computes κ_1 as follows:

$$\kappa_1 = (h^d)^{1/u} \cdot \left(\prod_{i=2}^{N} \kappa_i \right)^{-1}$$

Once the adversary provides the messages m_0 and m_1, $\mathcal{A}$ takes a random $\delta \in \{0,1\}$, samples random $r \in \mathbb{Z}_q$, and constructs the challenge ciphertext as described in the scheme:

$$CT^* = ((g^{su})^r, m_\delta \cdot (Z^{sv})^r) = (g^{sur}, m_\delta \cdot Z^{svr})$$

$\mathcal{A}$ runs adversary $\mathcal{B}$ to obtain the guess δ' and decides that $d = a \cdot v$ when $\delta = \delta'$. Note that when $d = a \cdot v$, the escrow shares of the custodians are valid, and they are capable of re-encrypting the challenge ciphertext into valid re-encryption shares (although the adversary cannot retrieve the underlying message through combination since he does not know the escrow authority secret key). By our initial assumption, $\mathcal{B}$ guesses δ correctly with non-negligible advantage ε, and hence, $\mathcal{A}$ solves the DDH problem in $\mathbb{G}_2$ with the same advantage. On the contrary, when d is random, m_δ is information-theoretically hidden, so the advantage of $\mathcal{B}$ is 0. Thus, the overall success probability of $\mathcal{A}$ is $\frac{1}{2} + \frac{\varepsilon}{2}$, and therefore, $\mathcal{A}$ wins the IND-CUST-CPA game with non-negligible advantage $\frac{\varepsilon}{2}$.

□

As a consequence of the previous two theorems, if we assume that the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$, that is, the SXDH assumption (see Section 3.1), then the EDS cryptosystem satisfies both the IND-EA-CPA and IND-CUST-CPA security notions.

6 Extension: Threshold Escrowed Decryption Scheme (TEDS)

An interesting enhancement over the proposed scheme is to construct a threshold variant. The escrow decryption capability of the previous scheme is all-or-nothing: all the corresponding custodians have to collaborate; otherwise, the escrow decryption protocol is incomplete and the escrow authority cannot retrieve the original message. However, in some situations, it could be interesting to tolerate that some custodians do not participate in the escrow decryption protocol, for example, in the event of a technical failure that prevents a custodian from performing re-encryptions.

In this section, we describe how to modify our original proposal to support a (t, N)-threshold escrowed decryption functionality. The idea is basically to share the user's secret value β/u between several escrow shares using Shamir's secret sharing in the exponent. Shamir's secret sharing guarantees that the secret is information-theoretically hidden if less than t shares of the secret are known, as described in Section 3.3. In our TEDS variant, this implies that if less than t custodians participate in the escrow decryption protocol, the escrow authority has zero information regarding the underlying message.

This extension only affects the Key Generation and the Escrow Decryption protocols of the previous proposal, which means that Encryption and Decryption remain the same. For this reason, we only need to describe the Key Generation and the Escrow Decryption protocols.

6.1 Description of the scheme

Key Generation. As in the basic scheme, the user U first selects random secrets $u, \beta \in \mathbb{Z}_q$, and computes g^u and g^β. Next, he samples random coefficients $f_1, f_2, ..., f_{t-1} \in \mathbb{Z}_q$ and defines the polynomial $f(x) \in \mathbb{Z}_q[x]$, so that:

$$f(x) = \beta/u + f_1 x + f_2 x^2 + ... + f_{t-1} x^{t-1}$$

That is, $f(0) = \beta/u$. Now, let $\mathcal{C}$ be the set of indices of the custodians selected by U that will receive a partial escrow share $\widetilde{\kappa}_i$. For simplicity, we can assume that $\mathcal{C} = \{1, ..., N\}$. Each partial escrow share $\widetilde{\kappa}_i$ is generated as follows:

$$\widetilde{\kappa}_i = (pk_{EA})^{f(i)}$$

In this case, the "shared secret" among custodians is $(pk_{EA})^{f(0)} = (h^a)^{\beta/u}$. In the same fashion as Shamir's secret sharing, it can be seen that using Lagrange's polynomial interpolation the secret $h^{a\beta/u}$ can be reconstructed from any set of t partial escrow shares. Let $\mathcal{I} \subseteq \mathcal{C}$ be a set of t indices. Reconstruction of the shared secret can be done as follows:

$$\prod_{i \in \mathcal{I}} \widetilde{\kappa}_i^{\lambda_i} = h^{a\beta/u}, \quad \text{where } \lambda_i = \prod_{j \in \mathcal{I}/\{i\}} \frac{j}{j-i} \tag{5}$$

The user sends $(g^u, g^\beta, \{\widetilde{\kappa}_i\}_{i=1}^N)$ to CA for certification, as in the original scheme. Now, the CA must ensure that the input is valid, and more importantly, that reconstruction can be achieved with *any* subset of t shares. A simple, brute-force approach is to reconstruct the shared secret with the $\binom{n}{t}$ possible

subsets of t shares out of n, and to verify first that the result is always the same, and, second, that the following equation holds:

$$e(g^u, \prod_{i \in \mathcal{I}} \widetilde{\kappa}_i^{\lambda_i}) = e(g^\beta, pk_{EA}) \tag{6}$$

The number $\binom{n}{t}$ of possible subsets can grow rapidly with n, so we describe in the Appendix an alternative non-exhaustive approach to check the validity of the shares. Instead of checking all possible subsets, the CA uses Lagrange's interpolating polynomial to check that the coefficients of the polynomial generated by the user are consistent with a polynomial of degree $t-1$ (that is, it can be reconstructed with t shares). However, we note that in a real deployment, n will probably be a low value (e.g., 5-8 custodians), so the brute-force approach suffices. The rest of the Key Generation protocol remains the same than in the previous scheme.

Escrow decryption. The first part of escrow decryption (i.e., the re-encryption share function) remains the same. Therefore, when the escrow authority asks custodian C_i, which has escrow share κ_i, for the re-encryption of a ciphertext $CT = (CT_1, CT_2)$, the custodian computes:

$$\rho_i = \mathsf{ShareReEnc}(CT_1, \kappa_i) = e(g^{sur}, \kappa_i)$$

After getting the re-encryption shares from t custodians (whose indices are contained in the set $\mathcal{I}$), the escrow authority executes the combination algorithm Comb to decrypt the ciphertext. In our proposal, this algorithm is defined as follows:

$$\mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i \in \mathcal{I}}) = \frac{CT_2}{(\prod_{i \in \mathcal{I}} \rho_i^{\lambda_i})^{1/sk_{EA}}}$$

6.2 Correctness

Since the public and private keys, and encryption and decryption functions remain the same, we only need to show the correctness of the key generation and escrow decryption procedures.

6.2.1 Key Generation

The arguments for correctness of the key generation procedure are very similar to those in the original scheme (see Section 5.2). The only difference is that reconstruction of the secret can be done with a subset of t shares, as opposed to the original scheme, which required all the shares; however, in both cases the reconstructed secret has the same form (i.e., $h^{a\beta/u}$), whose validity is checked with a pairing equation (Equation 6 in this scheme, and Equation 3 in the original).

6.2.2 Escrow Decryption

With respect to the escrow decryption procedure, first we note that the following relation holds from Equation 5:

$$\prod_{i \in \mathcal{I}} \kappa_i^{\lambda_i} = (\prod_{i \in \mathcal{I}} \widetilde{\kappa}_i^{\lambda_i})^\gamma = (h^{a\beta/u})^\gamma = h^{av/u}$$

During the Combination algorithm, the escrow authority computes the term $\prod_{i \in \mathcal{I}} \rho_i^{\lambda_i}$. By the bilinear property of the pairing, and the previous equation, it can be observed that the following holds:

$$\begin{aligned} \prod_{i \in \mathcal{I}} \rho_i^{\lambda_i} &= \prod_{i \in \mathcal{I}} e(g^{sur}, \kappa_i^{\lambda_i}) \\ &= e(g^{sur}, \prod_{i \in \mathcal{I}} \kappa_i^{\lambda_i}) \\ &= e(g^{sur}, h^{av/u}) = Z^{savr} \end{aligned}$$

Therefore, the Combination algorithm correctly retrieves the original message, since:

$$\begin{aligned}\mathsf{Comb}(sk_{EA}, CT, \{\rho_i\}_{i\in\mathcal{I}}) &= \frac{CT_2}{(\prod_{i\in\mathcal{I}} \rho_i^{\lambda_i})^{1/sk_{EA}}} \\ &= \frac{Z^{svr}\cdot m}{(Z^{savr})^{1/a}} = m\end{aligned}$$

6.3 Security analysis

The changes introduced in this variant with respect to the original are minor, and have almost no impact in the satisfaction of the security notions described in Section 4.5.

6.3.1 Security against the escrow authority

Theorem 3 (TEDS is IND-EA-CPA-secure)**.** *If the DDH problem in $\mathbb{G}_1$ is hard, then the TEDS cryptosystem is secure under the IND-EA-CPA notion.*

The public keys, secret keys, and Encryption and Decryption procedures are the same than in the previous scheme. In addition, the IND-EA-CPA notion does not allow the adversary to know the encryption shares, which are one of the few main differences between the TEDS and EDS schemes. Therefore, the same security analysis provided in Section 5.3 is applicable here.

6.3.2 Security against the custodians

Theorem 4 (TEDS is IND-CUST-CPA-secure)**.** *If the DDH problem in $\mathbb{G}_2$ is hard, then the TEDS cryptosystem is secure under the IND-CUST-CPA notion.*

The proof for Theorem 2 is applicable here with few changes. In this case, the security notion requires to release the escrow shares to the adversary. However, in the aforementioned proof the simulator has the knowledge necessary to correctly construct escrow shares, since he samples the secret u for the target user (as opposed to the previous proof, where the simulator does not know u). The changes in the proof are that now the simulator defines a secret polynomial $f(x)$ as in the scheme, but with $f(0) = 1/u$. He computes escrow shares as $\kappa_i = (h^d)^{f(i)}$. This ensures that reconstruction of the secret is possible, since for all sets $\mathcal{I}$ of t escrow shares, it holds that:

$$\prod_{i\in\mathcal{I}} \kappa_i^{\lambda_i} = \prod_{i\in\mathcal{I}} (h^d)^{\lambda_i f(i)} = (h^d)^{1/u}$$

It can be observed that when $d = a \cdot v$, then escrow shares (and the reconstructed secret) is correct as described in the scheme, so the adversary can perform re-encryptions of ciphertexts, just as custodians can. The rest of the proof remains the same.

7 Performance

In this section we present theoretical and experimental analysis of the performance of the proposed schemes. For the theoretical part, we analyze computational costs in terms of the main operations performed, which in this case are the exponentiation and the pairing. Since we are dealing with asymmetric pairings, which implies three different groups, the computational costs of exponentiations of group elements are identified separately; hence, for a fixed set of parameters, we denote as C_p to the cost of a pairing operation, C_{e_1} to an exponentiation in $\mathbb{G}_1$, C_{e_2} to an exponentiation in $\mathbb{G}_2$, and C_{e_T} to an exponentiation in $\mathbb{G}_T$. We ignore other minor costs, such as multiplications and inversions. Note, however, that in our solutions, the number of multiplications depends mainly on the number of custodians (denoted by N); we can safely assume that a realistic implementation will involve only a reduced number of custodians. For the TEDS scheme, an additional parameter t denotes the threshold in use.

The computation costs of both EDS and TEDS are shown in Table 1. It can be seen that the most common operations in both schemes have a moderate cost (e.g., encryption costs two exponentiations, while decryption one pairing), which is acceptable for most scenarios. One aspect that is important to mention is that, for the TEDS scheme, the costs associated to the Key Generation in the CA side include the brute-force verification procedure, which accounts for the $\binom{n}{t}t$ exponentiations in $\mathbb{G}_2$. However, this

Table 1: Computational costs of selected PRE schemes

Algorithm	EDS	TEDS
Key Generation (User)	$2C_{e_1} + 2C_{e_2}$	$2C_{e_1} + (N+1)C_{e_2}$
Key Generation (CA)	$3C_p + 2C_{e_1} + (N+1)C_{e_2}$	$3C_p + 2C_{e_1} + (\binom{n}{t}t + N + 1)C_{e_2}$
Encryption	$C_{e_1} + C_{e_T}$	$C_{e_1} + C_{e_T}$
Decryption	C_p	C_p
Escrow Decryption (Custodian)	C_p	C_p
Escrow Decryption (Escrow Agent)	C_{e_T}	$t\ C_{e_T}$

figure can be substituted by $N(N-t+1)$ exponentiations if the validation procedure presented in the Appendix is used. This latter procedure is more efficient for larger values of N (roughly, >10), although such values does not seem suitable for realistic applications.

With regards to the experimental analysis, we implemented a prototype of both schemes in C++ using the MIRACL SDK library [25] in order to make a quantitative assessment of their performance of our proposal. We selected the default Barreto-Naehrig (BN) curve provided by the library, which is suitable for asymmetric parings and that achieves 128 bits of security [26]. With regard to the execution environment, the tests were performed in a laptop with an Intel Core 2 Duo processor @ 2.66 GHz and 8 GB of RAM. Experiments were executed 100 times and we took the average for each operation.

Table 2: Experimental performance in ms. of the EDS and TEDS schemes ($N = 4$, $t = 3$)

Operation	EDS	TEDS
Key Generation (User)	0.536	1.129
Key Generation (CA)	42.674	44.961
Encryption*	0.612	0.603
Decryption*	14.176	14.587
Escrow Decryption (Custodian)*	14.118	14.438
Escrow Decryption (Escrow Agent)	0.587	1.782

Table 2 shows the results of our experiments, assuming 4 custodians ($N = 4$) and a threshold of 3 in the case of TEDS ($t = 3$). It can be seen that the operation that has the biggest cost is Key Generation in the CA side. This is something expected, since the CA has to bear the burden of checking the validity of the partial escrow shares that come form the user side; however, this process is only performed once per user, and it is reasonable to assume that the CA is prepared to assume such load. We make an important remark here: while this operation seems relatively costly, the rest of operations are much less expensive, and furthermore, their cost is constant irrespective of the number of custodians (except for user's key generation in the TEDS scheme, which also depends on this number). In particular, the timing of the encryption and decryption operations (which will be the most used functions in a real setting) only depend on the size of the pairing groups. We also note that operations marked with * are actually identical between EDS and TEDS, and hence, the small variations between them depend only on experimental factors.

Finally, we remark that it is possible to further optimize some of the computations such as exploiting the fact that most pairing operations have a fixed argument (e.g., [27] achieves a speed-up around 30%), and the use of multiexponentiations and multipairings.

8 Conclusions

In this paper we present two escrowed decryption schemes, which are public-key encryption schemes that allow an escrow authority to decrypt messages; the core characteristic of an escrowed decryption scheme is that this decryption is done *without escrowing users' secret keys*, but only the *decryption capability*, and that it requires the participation of third-party entities called custodians. In other words, the decryption capability is "*escrowed*" to the custodians.

In our proposed schemes we construct the escrow decryption capability using techniques from proxy re-encryption. The basic idea is to use the conventional PKE-based functions of the PRE scheme (i.e., encryption and decryption) as a regular PKE scheme, and to use the re-encryption function to define the escrowed decryption capability. Our solution, inspired by the Accountable Escrow Encryption scheme from [4], solves some of the problems that arise in said scheme, such as removing the possibility of a collusion of custodians, as well as halves communication costs, since the escrow decryption procedure only takes one round of queries to the custodians (instead of two in the original scheme).

We also propose a security model for escrowed decryption schemes, which considers security against three types of adversaries, namely the escrow authority, the custodians, and external entities. All of these models are based on the traditional indistinguishability notion (IND), although extended to our particular setting. Additionally, in this paper we focus only in indistinguishability against chosen-plaintext attacks (IND-CPA).

As future work, it is necessary to further formalize the security notions of this new cryptosystem. In particular, it is interesting to define an equivalent of CCA-security on this context, and to propose a solution that satisfies this notion. Another line of research is to reduce the trust requirements of the scheme, in particular with respect to the CA. In order to solve this, a possible extension is that the custodians have also a private key, so that the CA cannot do anything with the escrow shares.

Acknowledgments

This work was partly supported by the Spanish Ministry of Economy and Competitiveness through the projects SMOG (TIN2016-79095-C2-1-R) and PRECISE (TIN2014-54427-JIN). The first author is supported by a grant from the National Cybersecurity Institute (INCIBE).

References

[1] Wikipedia, "Investigatory Powers Act 2016 — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Investigatory%20Powers%20Act%202016&oldid=751118773`, 2016. [Online; accessed 24-November-2016].

[2] Wikipedia, "FBI–Apple encryption dispute — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=FBI%E2%80%93Apple%20encryption%20dispute&oldid=750915931`, 2016. [Online; accessed 24-November-2016].

[3] The Washington Post, "Obama administration's draft paper on technical options for the encryption debate." `http://t.co/YKtD9VEKSf`, 2015. [Online; accessed 28-March-2017].

[4] J. Liu, M. D. Ryan, and L. Chen, "Balancing societal security and individual privacy: Accountable escrow system," in *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*, pp. 427–440, IEEE, 2014.

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.

[6] S. Micali, "Fair public-key cryptosystems," in *Annual International Cryptology Conference*, pp. 113–138, Springer, 1992.

[7] Y. Frankel and M. Yung, "Escrow encryption systems visited: attacks, analysis and designs," in *Annual International Cryptology Conference*, pp. 222–235, Springer, 1995.

[8] J. Kilian and T. Leighton, "Fair cryptosystems, revisited," in *Annual International Cryptology Conference*, pp. 208–221, Springer, 1995.

[9] J. Kroll, E. Felten, and D. Boneh, "Secure protocols for accountable warrant execution," *See http://www. cs. princeton. edu/felten/warrant-paper. pdf*, 2014.

[10] M. Kohlweiss and I. Miers, "Accountable metadata-hiding escrow: A group signature case study," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 206–221, 2015.

[11] L. Xu, X. Wu, and X. Zhang, "Cl-pre: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 87–88, ACM, 2012.

[12] M. Jakobsson, "On quorum controlled asymmetric proxy re-encryption," in *International Workshop on Public Key Cryptography*, pp. 112–121, Springer, 1999.

[13] H.-Y. Lin and W.-G. Tzeng, "A secure erasure code-based cloud storage system with secure data forwarding," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 6, pp. 995–1003, 2012.

[14] H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, "Towards end-to-end secure content storage and delivery with public cloud," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pp. 257–266, ACM, 2012.

[15] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[16] M. S. Kiraz and O. Uzunkol, "Still wrong use of pairings in cryptography." Cryptology ePrint Archive, Report 2016/223, 2016. `http://eprint.iacr.org/`.

[17] J. A. Akinyele, C. Garman, and S. Hohenberger, "Automating fast and secure translations from type-i to type-iii pairing schemes," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1370–1381, ACM, 2015.

[18] M. Abe, F. Hoshino, and M. Ohkubo, "Design in type-i, run in type-iii: Fast and scalable bilinear-type conversion using integer programming," in *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pp. 387–415, 2016.

[19] G. Ateniese, J. Camenisch, and B. De Medeiros, "Untraceable rfid tags via insubvertible encryption," in *Proceedings of the 12th ACM conference on Computer and communications security*, pp. 92–101, ACM, 2005.

[20] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros, "Practical group signatures without random oracles.," *IACR Cryptology ePrint Archive*, vol. 2005, p. 385, 2005.

[21] D. Nuñez, I. Agudo, and J. Lopez, "Proxy re-encryption: Analysis of constructions and its application to secure access delegation," *Journal of Network and Computer Applications*, vol. 87, pp. 193–209, 2017.

[22] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Advances in Cryptology—EUROCRYPT'98*, pp. 127–144, 1998.

[23] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, Nov. 1979.

[24] V. Shoup and R. Gennaro, "Securing threshold cryptosystems against chosen ciphertext attack," *Journal of Cryptology*, vol. 15, no. 2, pp. 75–96, 2002.

[25] MIRACL, "MIRACL Crypto SDK." `https://milagro.apache.org/`.

[26] D. F. Aranha, P. S. Barreto, P. Longa, and J. E. Ricardini, "The realm of the pairings," in *Selected Areas in Cryptography–SAC 2013*, pp. 3–25, Springer, 2013.

[27] C. Costello and D. Stebila, "Fixed argument pairings," in *International Conference on Cryptology and Information Security in Latin America*, pp. 92–108, Springer, 2010.

Appendix

It is necessary to ensure that the set of N partial escrow shares $\widetilde{\kappa}_i$ provided by the user is correct, that is, that it allows reconstruction of the shared secret that later will enable the escrow decryption process. The problem is that reconstruction must succeed with any subset of t escrow shares, since this is a threshold cryptosystem.

Since the parameter N is fixed in advance, and the escrow shares are generated using an index i as input (that is, the input to the polynomial is not an arbitrary value), then the Lagrange basis polynomials can be defined in advance as shown in the next Equation, which produces polynomials of degree $N-1$ with known coefficients $\ell_{i,j}$:

$$\ell_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{x-j}{i-j} = \sum_{j=0}^{N-1} \ell_{i,j} \cdot x^j$$

Coefficients $\ell_{i,j}$ only depend on the choice of N, and hence, are the same for all users in the system once parameter N has been established. It can be shown that the Lagrange interpolating polynomial can be equivalently expressed as:

$$p(x) = \sum_{j=0}^{N-1} x^j \cdot (\sum_{i=1}^{N} \ell_{i,j} \cdot f(i)) \tag{7}$$

Note however that in our scheme the polynomial is hidden in the exponent. Thus, for a fixed j in Equation 7, the term $\sum_{i=1}^{N} \ell_{i,j} \cdot f(i)$ must be the coefficient of the reconstructed polynomial. Knowing this, we can use the $\ell_{i,j}$ coefficients to check whether a coefficient of the reconstructed polynomial is 0 or not, by using the following auxiliary function:

$$\mathsf{CheckCoefficient}(j) = \prod_{i=1}^{N} (\widetilde{\kappa}_i)^{\ell_{i,j}}$$

Since the polynomial is in the exponent, then if the coefficient is 0, the result will be 1, irrespective of the basis of the exponentiation. Otherwise (i.e., a non-zero coefficient), the result is an arbitrary value other than 1. Knowing this, one have to verify that the following conditions hold:

$$\begin{aligned} \forall j \in \{t, N-1\} \quad & \mathsf{CheckCoefficient}(j) \stackrel{?}{=} 1 \\ j = t-1 \quad & \mathsf{CheckCoefficient}(j) \stackrel{?}{\neq} 1 \end{aligned}$$