Non-interactive Distributed Encryption: A New Primitive for Revocable Privacy*

David Galindo
University of Luxembourg
Luxembourg
david.galindo@uni.lu

Jaap-Henk Hoepman
TNO
Groningen, the Netherlands
jaap-henk.hoepman@tno.nl
Radboud University Nijmegen
Nijmegen, the Netherlands
jhh@cs.ru.nl

ABSTRACT

In this paper we introduce and instantiate a new cryptographic primitive, called *non-interactive distributed encryption*, that allows a receiver to decrypt a ciphertext only if a minimum number of different senders encrypt the same plaintext. The new functionality can be seen as the dual of the functionality provided by threshold cryptosystems. It is shown that this primitive can be used to solve real-world problems balancing security and privacy needs. In particular it is used to solve the *canvas cutters problem* (introduced below), that might be of independent interest.

Categories and Subject Descriptors

D.4.6 [**OPERATING SYSTEMS**]: Security and Protection—*Cryptographic controls*; K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Security

Keywords

privacy, threshold cryptosystems, distributed encryption

1. INTRODUCTION

Privacy — sometimes loosely defined as the 'right to be let alone' [34] — is considered a fundamental human right in many societies. It is "essential for freedom, democracy, psychological well-being, individuality and creativity" [29].

Homeland security, on the other hand, is considered a political top priority. Unfortunately people see security and privacy as conflicting requirements. It is often believed that they are a "zero-sum" game [25], and that one cannot be achieved without sacrificing the other. As a consequence, given the current emphasis on homeland security, quite privacy invasive systems are being implemented today.

This paper tries to help re-balancing this situation, by showing that systems that truly respect our privacy can be built, while still satisfying security and law enforcement needs.

1.1 Motivating problems

Consider the following real-life example, that was communicated to us by Dutch law enforcement. So called "canvas cutters" are criminals that roam the parking places along highways looking for trucks with valuable content. They look inside these trucks by cutting their canvas. These criminals distinguish themselves from other road users by the very frequent visits of parking places along a single stretch of highway on a single day. To identify possible canvas cutters, one could set up Automatic Number Plate Recognition (ANPR) systems at the entry of each parking place, and search the resulting data stream for cars that enter multiple parking places along the same highway on a single day. Apart from identifying police cars (that have similar driving patterns), this should identify canvas cutters as well. Clearly this poses a privacy threat, as the number plate of each and every car visiting a parking place is recorded and retained.

Another motivating example is logging access to databases. To be able to investigate abuse transaction in a database must be logged. Recording the identity of the person performing the transaction in plaintext would be highly privacy invasive, as it stores all use of a database for all its users. This is disproportional if the only abuse foreseen is of people trying to link database records across several different databases, and if investigations will only focus on particular transactions that happened on certain records stored by a (not a priori fixed) subset of these logged databases.

In both cases it is desirable to protect the privacy of the users of the system, and to design the system in such a way that the identity of a user is only revealed if his actions exceed a certain threshold. Readers familiar with the subject will realise that, in a way, a generalisation of Chaum's [14] double spending prevention for electronic cash is required.

*This research is supported in part by the research program Sentinels as project 'Revocable Privacy' (10532). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

WPES'11, October 17, 2011, Chicago, Illinois, USA.

1.2 The need for a technical approach

It is necessary to realise that legal or regulatory attempts to restrict access are inadequate. Rules and regulations may change over time, allowing for more possibilities to gather information about people after the fact. Such "function creep" occurs frequently: once the system, in principle, allows certain ways to collect data, sooner or later government officials or law enforcement will ask for an extension of powers. Therefore, the solution must be found in limiting possibilities at the outset, through technical means, in the architecture and design of the system.

This line of reasoning follows the idea of "code as code" [23]. By embedding the rules, procedures and regulations into the implementation of the system, they cannot be changed after the fact. This guarantees that the only way to change the rules, and to gather more information, is through a complete redesign (and re-implementation) of the system.

1.3 Revocable privacy

In essence the idea of *revocable privacy* is exactly this: to design systems in such a way that no personal information is available unless a user violates the *pre-established* terms of service. Only in that case, his personal details (and when and how he violated the terms) are revealed to certain authorised parties. We have elaborated on the concept elsewhere [22], so give only the following informal definition in this paper.

Definition 1.1 (Revocable privacy) *We say that a system implements* revocable privacy *if the* architecture *of the system* guarantees *that personal data is revealed only if a* predefined rule *has been violated.*

An example of such a rule is that the same event happens in at least k different locations. To implement this rule, we introduce a new cryptographic primitive called *non-interactive distributed encryption*, that allows a receiver to decrypt a ciphertext only if a minimum number of different senders encrypt the same plaintext. This corresponds to k different senders observing the same event.

1.4 State of the art

Existing techniques can be applied to build revocable privacy systems. In fact, the basic idea of revocable privacy is certainly not a new one: back in 1988 Chaum *et al.* [14] proposed a scheme for off-line digital cash where double spending a coin would reveal the identity of the owner of the coin. More recent examples are limiting the amount one can spend at a single merchant while remaining anonymous [12], or revoking anonymity of users that do not pay their bill at a merchant [10].

Similar techniques have been used to implement k-times anonymous authentication (k-AA) schemes, that allow a user to prove ownership of a credential anonymously, but at most k times [32, 2, 11]. See Section 6 for a discussion on the limitations of using k-times anonymous authentication schemes for solving the canvas cutters problem.

The PhD thesis of Marcus Stadler [30] from 1996 on revocable privacy is a first attempt to create a toolbox of cryptographic primitives for revocable privacy, providing several primitives. Fair blind signatures [31, 1] for example allow a signer to sign a message, such that the signer cannot later link its signature to the message it signed. In essence it corresponds to "blindly" signing a document. Fairness allow the signer to recover this link (possibly revoking privacy in this case), but only with the help of a trusted third party. Publicly verifiable secret sharing [15] allow all parties to verify that the inputs sent by the different parties are indeed properly distributed but without actually revealing that information (cf. [33] for an example that applies to key escrow in communication networks).

1.5 Organization

In Section 2 we introduce our new non-interactive distributed encryption primitive and present a realization of that primitive in Section 3. We show how that primitive can be made forward secure in Section 4. Section 5 describes how our primitive can be used to solve our motivating problems. We finish in Section 6 with a brief discussion of our results, and outline our plans for further research.

2. NON-INTERACTIVE DISTRIBUTED ENCRYPTION

Let us first describe threshold cryptosystems as a related, but existing, primitive. A k-out-of-n *threshold cryptosystem* [17, 27] distributes the task of non-interactively *decrypting* an encrypted message over a group of n users such that the task can be successfully completed if at least $1 < k \leq n$ honest users cooperate. More specifically, there are n receivers $R_1, \ldots, R_n$ that are sent encrypted content by a sender S. Key generation produces an encryption key E and a set of n corresponding decryption keys $D_1, \ldots, D_n$. To encrypt a plaintext p, a sender S uses the encryption key E and randomness r to produce a ciphertext $C = \mathbf{enc}(E, p; r)$. To decrypt an encrypted message at least k receivers are needed to locally and non-interactively compute decryption shares $C_i = \mathbf{share}(D_i, C)$. A so called combiner collects these decryption shares to produce the final plaintext p.

We initiate here the study of the dual of this primitive, that we call *non-interactive distributed encryption*. In such a scheme, a group of n senders $S_1, \ldots, S_n$, each of them holding an encryption key E_i, independently and without interaction encrypt messages to a combiner. We call a *ciphertext share* the encryption $c_i = \mathbf{Enc}(E_i, p; r_i)$ of a plaintext p with randomness r_i produced by any sender S_i. From these ciphertext shares a plaintext p is revealed if and only if the receiver obtains at least k ciphertext shares corresponding to p, each created by a different sender. For ease of presentation we will refer to "non-interactive distributed encryption" simply as "distributed encryption" (DE). We would like to point out that our investigations have shown that non-interactiveness is hard to achieve in an actual constructions. Primitives with similar properties as ours, like k-times anonymous authentication [11] and others, do exist but are all interactive.

The fact that threshold cryptosystems (TC) and DE are dual to each other raises the question of whether the two primitives are simply different formulations of the same functionality. But a closer looks shows that there are subtle yet significant differences between the two primitives.

1. While threshold cryptosystems in the public key setting are known to exist, it is easy to see that *public-key* non-interactive distributed encryption schemes can not be secure. In fact, let the encryption keys $E_1, \ldots, E_n$ be public and thus known to the combiner. In this

case a curious combiner can build ciphertext shares by itself. This implies that if the combiner suspects that a subset of $k-r$ ciphertext shares contain a certain plaintext p, then it suffices for him to build r extra encryptions of p under different unused public keys to test whether his guess is indeed correct. The conclusion is therefore that public key distributed encryption schemes can not attain the semantic security (see Definition 2.1) that we want to achieve.

2. The combine algorithms for TC and DE are of different nature. The combine algorithm for TC works under the assumption that the decryption shares are all produced using as input the *same random representation* of the plaintext p, that is $C = \mathbf{enc}(E, p; r)$. In sharp contrast, the combine algorithm in DE gets as input *independent random representations* of the same plaintext p, that is $c_i = \mathbf{Enc}(E_i, p; r_i)$ for random r_i's.

3. In DE there is no proper decryption algorithm and thus no proper decryption key. It is however certainly possible to use *additional* secure channels between the senders and the combiner when applying DE in a certain context. In such cases the combiner in effect does have a decryption key (see also section 5).

4. DE schemes actually provide a powerful functionality that can be seen as a sort of secret sharing scheme that, after the distribution of some setup values (the individual encryption keys), allows n parties to compute shares of arbitrary values p without interaction. We do not know of any secret sharing scheme in the literature enjoying this feature. Actually, the primitive with the closest functionality that we can think of is a non-interactive threshold pseudorandom function, for which very few constructions are known [19, 9].

All the above suggests that it is difficult to construct a DE scheme from a TC, and in fact we have not found any such construction.

A Misnomer. We have to point out an unfortunate misnomer regarding threshold cryptosystems. Some recent papers, e.g., [6, 16, 35], refer to threshold cryptosystems as threshold encryption. But in fact *threshold decryption* (the term is used in [3]) would have been a more appropriate name for this primitive, given the fact that any party can produce a valid encryption all by itself and that what is being distributed is the decryption operation. We would have preferred to use the term threshold encryption for our new primitive. However we have decided to use the more general term "distributed encryption" to prevent confusion.

Related Work. A related primitive is *shared encryption for block ciphers* proposed by Martin *et al.* [24]. In shared encryption the receiver must succeed in decrypting the ciphertext if it has been produced by an authorised subset and detect a forgery otherwise. Contrary to our approach, Martin *et al.* concentrate on *interactive* shared encryption schemes, which makes the problem less challenging. In fact in their schemes interaction allows senders to sequentially produce a ciphertext, each sender using its own key material, in a way similar to onion-routing [13]. Therefore the combiner only receives the final ciphertext. In our framework, senders must locally and independently produce the ciphertext shares. Moreover, in contrast to [24], we are able to construct a scheme with constant-size ciphertexts and encryption keys (cf. Section 3).

2.1 Syntax

Formally, a k-out-of-n *distributed encryption* scheme DE consists of three algorithms $\mathrm{DE} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Comb})$:

Gen$(1^\ell, k, n)$ Given a security parameter 1^ℓ, number of users n and threshold parameter k, this function generates encryption keys $E_1, \ldots, E_n$. Additionally it outputs the description of plaintext and ciphertext spaces $\mathcal{P}$ and $\mathcal{C}$. A plaintext p is called admissible if $p \in \mathcal{P}$; admissible ciphertexts are defined analogously.

Enc(E_i, p) Given an encryption key E_i corresponding to sender i and a plaintext p, this function returns a ciphertext share c_i.

Comb(C) Given a set of ciphertext shares $C = \{c_{i_1}, \ldots, c_{i_k}\}$ with cardinality k, **Comb**(C) either returns a plaintext p or error.

Every distributed encryption scheme must satisfy the following correctness requirement.

Correctness Let $E_1, \ldots, E_n$ be the encryption keys obtained by running the key generation algorithm. Let $C = \{c_{i_1}, \ldots, c_{i_k}\}$, where $c_{i_t} = \mathbf{Enc}(E_{i_t}, p_{i_t})$ for sender i_t and plaintext p_{i_t}. Then, with overwhelming probability, **Comb**(C) returns p iff $p_{i_1} = \ldots = p_{i_k} = p$ and $i_1, \ldots, i_k$ are pairwise different.

2.2 Security definition

The communication model of distributed encryption is summarized as follows.

1. The encryption keys $E_1, \ldots, E_n$ are secret, meaning that each encryption key E_i is only known to the legitimate sender S_i.

2. Senders produce their ciphertexts locally and independently from other senders.

3. Only the combiner gets to see the ciphertext shares produced by the different senders [1].

Roughly speaking, the security of a distributed encryption scheme is defined as follows: a combiner that corrupts r senders and learns their secret encryption keys $E_{i_1}, \ldots, E_{i_r}$, should gain (almost) no information on any plaintext p contained in a set of $k-1-r$ ciphertext shares produced by non-corrupted senders.

We proceed to give the formal security definition of a distributed encryption scheme DE. We start by stating the access that an adversary $\mathcal{A}$, that plays the role of a curious combiner and that can corrupt a subset of senders, has to the system:

1. We give the adversary access to an encryption oracle $\mathcal{O}^{\mathbf{E}}(\cdot,\cdot)$ that on input (i, p), where $i \in \{1, \ldots, n\}$ and p is an admissible plaintext returns $\mathbf{Enc}(E_i, p)$. We do not need to provide the adversary with access to any decryption oracle since, as discussed before, in a distributed encryption scheme there are no proper decryption keys.

[1] This is trivially implemented by giving each sender S_i an individual encryption key k_i, shared with the combiner, for a semantically-secure symmetric encryption scheme.

2. The adversary is allowed to corrupt r senders up to a threshold $r < k$, meaning that it learns their local state (e.g. the secret encryption keys). We need to distinguish between *static corruptions*, where the adversary must decide which senders it wants to corrupt before the execution of the protocol, and *adaptive corruptions*, where the adversary can corrupt players during the execution of the protocol.

3. Finally the adversary is challenged on $k-1-r$ ciphertext shares that were produced by non-corrupted senders and that all correspond to the same plaintext p.

Next we set the confidentiality goal against such an adversary. We will do so by extending the classical notion of *indistinguishability of ciphertexts* [21] to our setting. Roughly speaking, this requires that an adversary is unable to distinguish between encryptions of adversarially chosen plaintexts p_0 and p_1.

The security notion *indistinguishability against chosen-plaintext attacks*, referred to as IND, is obtained by combining the above confidentiality goal and attack model.

Definition 2.1 (IND) *Let us consider a* k-out-of-n *distributed encryption scheme* $\mathrm{DE} = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Comb})$. *We define the following game between a challenger and an adversary* $\mathcal{A}$*:*

Phase 0 (Only for static adversaries) *The adversary outputs a set of* r *indexes* $I_c = \{i_1, \ldots, i_r\} \subset \{1, \ldots, n\}$ *and* $0 \leq r < k$. *An index* $i \in I_c$ *denotes that the adversary corrupts sender* S_i. *It also outputs* $k-1-r$ *pairwise different indexes* $I_{nc} = \{i_{r+1}, \ldots, i_{k-1}\}$ *corresponding to the non-corrupted senders whose ciphertext shares are to appear in the challenge phase.*

Setup *The challenger runs* $(E_1, \ldots, E_n) \leftarrow \mathbf{Gen}(1^\ell, k, n)$.

Find *The set of queried plaintexts* Q *is initialized to* $\varnothing$. *For adaptive adversaries the set* I_c *of corrupted senders is initialized to* $I_c = \varnothing$.

The adversary can issue two types of queries:

- *On encryption queries of the form* $\mathsf{enc}(i, p)$, *where* $i \in \{1, \ldots, n\}$, $i \notin I_c$, *and* p *is an admissible plaintext, the adversary receives* $\mathbf{Enc}(E_i, p)$. *Moreover,* p *is added to* Q.
- (Only for adaptive adversaries) *On corruption queries* $\mathsf{corrupt}(I_{tbc})$, *where* $I_{tbc} \subset \{1, \ldots, n\}$ *is a possibly non-empty set, the challenger proceeds as follows:*
  - *for all* $i \in I_{tbc}$, *it adds* i *to* I_c. *If* $|I_c| \geq k$ *then the game aborts and the adversary looses.*
  - *for all* $i \in I_c$ *the adversary receives* E_i.

The cardinality of I_c *at the end of the* Find phase *is denoted by* r, *and it holds that* $r < k$ *if the game was not aborted.*

Challenge $\mathcal{A}$ *outputs two equal-length plaintexts* $p_0, p_1 \in \mathcal{P}$ *such that* $p_0, p_1 \notin Q$. *Additionally, in the case of adaptive adversaries,* $\mathcal{A}$ *also outputs* $k-1-r$ *indexes* $I_{nc} = \{i_{r+1}, \ldots, i_{k-1}\}$ *corresponding to the senders on which* $\mathcal{A}$ *wants to be challenged. Naturally, it is required that none of the senders on which* $\mathcal{A}$ *wants to be challenged have been corrupted. Next, the challenger chooses a random bit* β *and returns* $\{c_i \mid i \in I_{nc}\}$ *where* $c_i = \mathbf{Enc}(E_i, p_\beta)$.

Guess *The adversary* $\mathcal{A}$ *outputs a guess* $\beta' \in \{0, 1\}$. *The adversary wins the game if* $\beta = \beta'$.

Define $\mathcal{A}$*'s advantage as* $\mathbf{Adv}^{\mathrm{IND}}_{\mathrm{DE},\mathcal{A}}(1^\ell) = \left|\Pr[\beta' = \beta] - 1/2\right|$. *A scheme* DE *is said to have* indistinguishability of ciphertexts (IND secure) *if* $\mathbf{Adv}^{\mathrm{IND}}_{\mathrm{DE},\mathcal{A}}(1^\ell)$ *is negligible for every PPT adversary* $\mathcal{A}$.

Remark 2.2 The reason why we only allow $\mathcal{A}$ to receive $k-1-r$ ciphertext share challenges is the following. The adversary can construct r ciphertext shares Y_0 corresponding to encryption of the plaintext p_0 under the corrupted keys (and similarly the set Y_1 for p_1). If the adversary is given $k-r$ ciphertext share challenges $X = \{c_i \mid i \in I_{nc}\}$ then the combined set $X \cup Y_\beta$ has size k and can trivially be given to $\mathbf{Comb}(\cdot)$ to see if it returns p_β.

3. OUR SCHEME

In this section we propose an efficient non-interactive distributed encryption scheme by combining the Boneh-Franklin threshold identity-based encryption scheme and (symmetric) authenticated encryption. We start by describing some building blocks.

3.1 Preliminaries

In the following we recall the definitions of the Decisional Bilinear Diffie-Hellman (BDDH) and the Decisional Diffie-Hellman assumptions, Lagrange coefficients for polynomial interpolation and one-time secure authenticated encryption.

Definition 3.1 (Asymmetric Pairing Groups) *Let* $\mathbf{G}_1 = \langle g_1 \rangle$, $\mathbf{G}_2 = \langle g_2 \rangle$ *and* $\mathbf{G}_T$ *be (cyclic) groups of order* q *prime. A map* $\mathbf{e} : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$ *to a group* $\mathbf{G}_T$ *is called a* bilinear *map, if it satisfies the following two properties:*

Bilinearity: $\mathbf{e}(g_1^a, g_2^b) = \mathbf{e}(g_1, g_2)^{ab}$ *for all integers* a, b

Non-Degenerate: $\mathbf{e}(g_1, g_2)$ *has order* q *in* $\mathbf{G}_T$.

We assume there exists an efficient bilinear pairing instance generator algorithm $\mathcal{IG}$ *that on input a security parameter* 1^ℓ *outputs the description of* $\langle \mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, q \rangle$, *with* q *a* ℓ*-bit length prime.*

Asymmetric pairing groups can be efficiently generated [4, 20]. Pairing croups exponentiations and pairing operations can also be efficiently computed [18].

There are several flavors of the BDDH and Decisional Diffie-Hellman assumptions over asymmetric pairings [28]. The formulations we give here are motivated by our security reduction.

Definition 3.2 (DDH1 assumption) *Let us define*

$$\mathbf{Z} \leftarrow \left(\mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, q, g_1, g_2, g_1^a, g_1^b\right)$$

where $\langle \mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, q\rangle \leftarrow \mathcal{IG}(1^\ell)$ *and* $a, b \xleftarrow{\$} \mathbf{Z}_q^*$. *We say that* $\mathcal{IG}$ *satisfies the* Decisional Diffie-Hellman assumption in $\mathbf{G}_1$ *if for* $r \xleftarrow{\$} \mathbf{Z}_q^*$ *the value*

$$\mathbf{Adv}^{\mathrm{DDH1}}_{\mathcal{IG},\mathcal{A}}(\ell) := \left| \boldsymbol{Pr}\left[\mathcal{A}(\mathbf{Z}, g_1^{ab}) = 1\right] - \boldsymbol{Pr}[\mathcal{A}(\mathbf{Z}, g_1^{r}) = 1] \right|$$

is negligible in ℓ. *The probabilities are computed over the internal random coins of* $\mathcal{A}$, $\mathcal{IG}$ *and the random coins of the inputs.*

Definition 3.3 (BDDH assumption) *Let us define*

$$\mathbf{Z} \leftarrow \left(\mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, q, g_1, g_2, g_1^a, g_1^b, g_1^c, g_2^b, g_2^c\right)$$

where $\langle \mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, q\rangle \leftarrow \mathcal{IG}(1^\ell)$ *and* $a, b, c \xleftarrow{\$} \mathbf{Z}_q^*$. *We say that* $\mathcal{IG}$ *satisfies the* Decisional Bilinear Diffie-Hellman assumption *if for* $r \xleftarrow{\$} \mathbf{Z}_q^*$ *the value*

$$\mathbf{Adv}^{\mathrm{BDDH}}_{\mathcal{IG},\mathcal{A}}(\ell) :=$$
$$\left| \boldsymbol{Pr}\left[\mathcal{A}(\mathbf{Z}, \mathbf{e}(g_1,g_2)^{abc}) = 1\right] - \boldsymbol{Pr}[\mathcal{A}(\mathbf{Z}, \mathbf{e}(g_1,g_2)^{r}) = 1] \right|$$

is negligible in ℓ. *The probabilities are computed over the internal random coins of* $\mathcal{A}$, $\mathcal{IG}$ *and the random coins of the inputs.*

Definition 3.4 (Lagrange coefficients) *For a key reconstruction set* $\mathcal{I} \subseteq \{1,\ldots,n\}$ *we define the* Lagrange Coefficients $\lambda_i^{\mathcal{I}}$ *as* $\lambda_i^{\mathcal{I}} = \prod_{t\in\mathcal{I}\setminus\{i\}} \frac{t}{t-i} \in \mathbf{Z}_q^*$. *For any polynomial* $P \in \mathbf{Z}_q[X]$ *of degree at most* $|\mathcal{I}|-1$ *this entails* $\sum_{i\in\mathcal{I}} P(i)\lambda_i^{\mathcal{I}} = P(0)$.

Definition 3.5 (Authenticated Encryption) *Formally, a* symmetric authenticated encryption *scheme* AE *consists of three algorithms* $\mathrm{AE} = (\mathbf{gen}, \mathbf{enc}, \mathbf{dec})$*:*

$\mathbf{gen}(1^\ell)$ *Given a security parameter* 1^ℓ *this function generates a secret key* $K \xleftarrow{\$} \mathbf{G}_T$ *(tailored to our setting). Optionally it outputs a plaintext space* $\mathcal{P}_{\mathrm{AE}}$.

$\mathbf{enc}(K, p)$ *Given a secret key* $K \in \mathbf{G}_T$ *and a plaintext* p, *returns a ciphertext* α.

$\mathbf{dec}(K, \alpha)$ *Given a secret key* $K \in \mathbf{G}_T$ *and a ciphertext* α, *it returns a plaintext* p *(if and only if* $\alpha = \mathbf{enc}(K, p)$*) or an error symbol* error.

Definition 3.6 (One-time secure AE**)** *We require the* AE *scheme to provide* privacy and authenticity against one-time attacks*, that is, encryption keys are one-time. This is captured by the following game between a challenger and an adversary* $\mathcal{A}$*:*

Setup *The challenger runs* $(K, \mathcal{P}_{\mathrm{AE}}) \leftarrow \mathbf{gen}(1^\ell)$.

Find *The adversary outputs two equal-length admissible messages* $p_0, p_1 \in \mathcal{P}_{\mathrm{AE}}$.

Challenge *The challenger chooses a random bit* β *and returns* $\alpha^\star = \mathbf{enc}(K, p_\beta)$. *The adversary is allowed to submit a single decryption query* $\alpha \neq \alpha^\star$. *If* $\beta = 0$ *the challenger answers* $\mathbf{dec}(K, \alpha)$*; if* $\beta = 1$ *the challenger answers* error.

Guess *The adversary* $\mathcal{A}$ *outputs a guess* $\beta' \in \{0,1\}$. *The adversary wins the game if* $\beta = \beta'$.

Define $\mathcal{A}$*'s advantage as* $\mathbf{Adv}^{\mathrm{ae-ot}}_{\mathrm{AE},\mathcal{A}}(1^\ell) = \left|\Pr[\beta' = \beta] - 1/2\right|$. *An authenticated encryption scheme* AE *is called* one-time secure *if* $\mathbf{Adv}^{\mathrm{ae-ot}}_{\mathrm{AE},\mathcal{A}}(1^\ell)$ *is negligible for every PPT adversary* $\mathcal{A}$.

One-time secure AE schemes can be efficiently and generically build from (one-time secure) IND-CPA symmetric encryption and message authentication codes [5].

3.2 The scheme

The intuition behind it is as follows. Consider the Boneh-Franklin (BF) identity-based encryption scheme with threshold user-key generation [7] (Section 6) over asymmetric pairings. The master public key consists of $\Gamma = g_2^e \in \mathbf{G}_2$ and a hash function $H : \{0,1\}^* \to \mathbf{G}_1$. Let the master secret key $e \in \mathbf{Z}_q^*$ be shared using a k-out-of-n Shamir's secret sharing scheme [26], resulting in shares $e_1, \ldots, e_n \in \mathbf{Z}_q^*$. The encryption keys are given as $E_i = (\Gamma, e_i)$ for $i \in \{1,\ldots,n\}$. A ciphertext share on plaintext p by sender S_i consists of two parts. The first part is obtained by encrypting plaintext p under identity p using the IND-CPA version of Boneh-Franklin scheme [7] (Section 4.1). The second part is produced by computing the private-key share η_i corresponding to the identity p using the i-th share e_i. Once k ciphertext shares corresponding to pairwise different senders are available, decryption proceeds by reconstructing the private-key η from private-key shares $\eta_{i_1}, \ldots, \eta_{i_k}$, and then decrypting the BF ciphertext with the obtained decryption key. If all ciphertext shares correspond to the same plaintext p, then a genuine private-key $\eta = H(p)^e \in \mathbf{G}_1$ is obtained, and decryption of the BF ciphertext will return the plaintext p.

Here follows the description of our scheme. Let $\mathrm{AE} = (\mathbf{gen}, \mathbf{enc}, \mathbf{dec})$ be a one-time secure authenticated encryption scheme and let $\mathcal{IG}(\cdot)$ be a pairing generator algorithm. Starting from these primitives, we build a k-out-of-n distributed encryption scheme as follows:

$\mathbf{Gen}(1^\ell, k, n)$ First generate an asymmetric pairing, by running

$$\langle \mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, q\rangle \leftarrow \mathcal{IG}(1^\ell).$$

Let $H : \{0,1\}^* \to \mathbf{G}_1$ be a hash function mapping strings to elements in $\mathbf{G}_1$ (see [8] for an efficient implementation of such a hash function). Generate a master secret key $e \xleftarrow{\$} \mathbf{Z}_q^*$. Define $\Gamma = g_2^e \in \mathbf{G}_2$ to be the corresponding public key. Share the secret key using Shamir's k-out-of-n secret sharing by choosing $\epsilon_1, \ldots, \epsilon_{k-1} \xleftarrow{\$} \mathbf{Z}_q^*$ and defining the $k-1$ degree polynomial $P(x) = e + \sum_{t=1}^{k-1} \epsilon_t \cdot x^t$. For $i \in \{1,\ldots,n\}$ set $e_i = P(i) \in \mathbf{Z}_q^*$ and let $E_i = (i, H, \Gamma, e_i)$. The space of plaintexts $\mathcal{P}$ is set to be $\mathcal{P}_{\mathrm{AE}}$, where $\mathcal{P}_{\mathrm{AE}} \leftarrow \mathrm{AE}.\mathbf{gen}(1^\ell)$.

$\mathbf{Enc}(E_i, p)$ Given an encryption key E_i and a plaintext $p \in \mathcal{P}$, choose $r \xleftarrow{\$} \mathbf{Z}_q^*$. Now compute $K \leftarrow \mathbf{e}(H(p), \Gamma)^r$ and set

$$\eta_i = H(p)^{e_i}, \quad \gamma_i = g_2^r, \quad \alpha_i = \mathrm{AE}.\mathbf{enc}(K, p)$$

Return $c_i = \langle i, \eta_i, \gamma_i, \alpha_i\rangle$.

$\mathbf{Comb}(C)$ Parse C as $\{c_{i_1}, \ldots, c_{i_k}\}$. Parse each $c_{i_t} \in C$ as

$\langle i_t, \eta_{i_t}, \gamma_{i_t}, \alpha_{i_t}\rangle$, construct[2] $\mathcal{I} = \{i_1, \ldots, i_k\}$ and compute $\lambda_t^{\mathcal{I}}$ for all $t \in \mathcal{I}$. Compute $h \leftarrow \prod_{t\in\mathcal{I}} (\eta_t)^{\lambda_t^{\mathcal{I}}}$, then compute $K \leftarrow \mathbf{e}(h, \gamma_{i_1})$ and return AE.$\mathbf{dec}(K, \alpha_{i_1})$.

Correctness Let $E_1, \ldots, E_n$ be the encryption keys obtained by running the key generation algorithm. Let

$$C = \{c_{i_1}, \ldots, c_{i_k}\},$$

where $c_i = \mathbf{Enc}(E_i, p)$ and $i_1, \ldots, i_k$ are pairwise different. We want to see that $\mathbf{Comb}(C) = p$. Indeed, let $\mathcal{I} = \{i_1, \ldots, i_k\}$. Since the ciphertext shares are correctly computed, then we have that

$$h = \prod_{t\in\mathcal{I}} (\eta_t)^{\lambda_t^{\mathcal{I}}} = \prod_{t\in\mathcal{I}} H(p)^{e_t \lambda_t^{\mathcal{I}}} = \prod_{t\in\mathcal{I}} H(p)^{P(t)\lambda_t^{\mathcal{I}}} =$$
$$= H(p)^{\sum_{t\in\mathcal{I}} P(t)\lambda_t^{\mathcal{I}}} = H(p)^{P(0)} = H(p)^{e}.$$

By setting $K \leftarrow e(H(p)^e, \gamma_{i_1})$ we obtain that $\mathbf{dec}(K, \alpha_{i_1}) = p$, since if $\gamma_{i_1} = g_2^r$ then $\mathbf{e}(H(p)^e, g_2^r) = \mathbf{e}(H(p)^r, g_2^e) = \mathbf{e}(H(p), \Gamma)^r$.

Let us now see that when $c_i = \mathbf{Enc}(E_i, p)_{i,p}$, with $i \in \{i_1, \ldots, i_k\}$ and $p \in \{p_1, \ldots, p_k\}$, and the conditions $p_1 = \ldots = p_k = p$ are not satisfied, then $\mathbf{Comb}(C)$ returns error with overwhelming probability. Indeed, let us assume there exists $2 \le j \le k$ such that $p_1 \neq p_j$. Let us consider the encryption key K' computed when combining the shares as

$$K' \leftarrow \mathbf{e}(h, \gamma_{i_1}) \text{ where } h = \prod_{t\in\{1,\ldots,k\}} H(p_t)^{e_{i_t}\lambda_{i_t}}.$$

It is easy to see that, unless a collision occurs in H, the key K' is uniformly at random distributed in $\mathbf{G}_T$ and independently from the genuine encryption key K used to compute c_{i_1}. This is a consequence of H being modelled as a random oracle, which firstly implies that collisions in H only happen with negligible probability; secondly it implies that $H(p_1)$ and $H(p_j)$ are independently and uniformly distributed for $p_1 \neq p_j$. It only lacks to see that $\mathbf{dec}(K', \mathbf{enc}(K, p_1))$ returns error with overwhelming probability. Indeed, the one-time security of AE (cf. Definition 3.6) implies that it is infeasible for an adversary to come up with any valid ciphertext α not returned by $\mathbf{enc}(K', \cdot)$, which proves the correctness of the scheme.

Remark 3.7 Let us briefly discuss why the assumption that DDH1 is hard is needed for the IND security proof. If an adversary $\mathcal{A}$ queries $\mathcal{O}^{\mathbf{E}}(i, p)$ for $i \in \{1, \ldots, n\}$ and p an admissible plaintext, it will get back a ciphertext share of the form $(H(p)^{e_i}, g_2^r, \mathbf{enc}(K, p))$. Let $p_0 \neq p_1 \neq p$ be the plaintexts on which $\mathcal{A}$ wants to be challenged. Now the challenge ciphertext will have the form $\left(H(p_\beta)^{e_i}, g_2^{r'}, \mathbf{enc}(K_\beta, p_\beta)\right)$ for $\beta \xleftarrow{\$} \{0,1\}, r' \xleftarrow{\$} \mathbf{Z}_q$. Then $\mathcal{A}$ can verify whether a guess $\beta' \in \{0,1\}$ is correct by using a DDH1 solver (whose existence is guaranteed if the DDH1 assumption is not satisfied) on input $(H(p), H(p_{\beta'}), H(p)^{e_i}, H(p_\beta)^{e_i})$. In effect, let us write $g_1 := H(p)$, $g_1^a := H(p_{\beta'})$, $b := e_i$. Then $H(p_\beta)^{e_i} = g_1^{ab}$ iff $\beta = \beta'$.

Result 3.8 *Let $\mathcal{A}$ be an adversary in the* static corruptions *scenario against the* IND *security of the above k-out of-n* DE *scheme. Assume H is modeled as a random oracle and that $\mathcal{A}$ makes at most q_H queries. Then there exist algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ such that*

$$\mathbf{Adv}^{\mathrm{IND}}_{\mathrm{DE},\mathcal{A}}(1^\ell) \le \tfrac{q_H}{2}\big(k-1\big)\big(\mathbf{Adv}^{\mathrm{BDH}}_{\mathcal{IG},\mathcal{B}_2}(1^\ell) + \mathbf{Adv}^{\mathrm{DDH1}}_{\mathcal{IG},\mathcal{B}_1}(1^\ell)\big) + \tfrac{q_H}{2}\mathbf{Adv}^{\mathrm{ae-ot}}_{\mathrm{AE},\mathcal{B}_3}(1^\ell)$$

PROOF. We provide a game-based proof of the previous lemma. We will make use of the following simple "Difference Lemma".

Lemma 3.9 *Let Y_1,Y_2, B be events defined in some probability distribution, and suppose that $Y_1 \wedge \neg B \Leftrightarrow Y_2 \wedge \neg B$. Then $|\Pr[Y_1] - \Pr[Y_2]| \le \Pr[B]$.*

The proof of the Result 3.8 is obtained by considering subsequent games, Game 0 to Game 6. These games are produced by a simulator and are obtained successively from slight modifications of the previous game. It starts with Game 0, which is the original IND game. In every game the simulators' output bit β' will be well-defined. For $0 \le i \le 6$ we define the event

$$X_i : \text{The simulator outputs } \beta' = \beta \text{ in Game } i.$$

In Game 0 we let the simulator output the same value for β' as output by the IND adversary $\mathcal{A}$. Then, since in Game 0 the simulator exactly plays the IND security experiment with adversary $\mathcal{A}$, we have $|\Pr[X_0] - 1/2| = \mathbf{Adv}^{\mathrm{IND}}_{\mathrm{DE},\mathcal{A}}(1^\ell)$.

Let q_H be an upper bound on the number of pairwise different queries to the random oracle H made by the adversary. Let q_E be the number of adversarial encryption queries. Let us assume without loss of generality that $\{1, \ldots, r\}$ is the set of corrupted senders, $\{r, \ldots, k-1\}$ the set of senders' indexes on which the adversary wants to be challenged and $\{k, \ldots, n\}$ the set of honest senders. Let

$$\Big(\mathbf{e}(\cdot,\cdot), \mathbf{G}, \mathbf{G}_T, q, g_1, g_2, g_1^a, g_1^b, g_1^c, g_2^b, g_2^c, T\Big)$$

be the input of the BDDH problem, where $T \leftarrow \mathbf{e}(g,g)^{abc}$ or $T \xleftarrow{\$} \mathbf{G}_T$.

Game 1 (Change generation of encryption keys) In this game we change the generation of the initial encryption keys $E_1, \ldots, E_n$ by using the input of the BDDH problem as follows:

1. (Defining $e_1, \ldots, e_{k-1}$) Pick $e_1, \ldots, e_{k-1} \xleftarrow{\$} \mathbf{Z}_q$. Let $P \in \mathbf{Z}_q[X]$ of degree $k-1$ be the unique polynomial implicitly defined by $P(0) = b$ and $P(i) = e_i$ for $1 \le i \le k-1$. The simulator computes $h_i = g_1^{e_i}$ for $1 \le i \le k-1$. Note that the simulator does not explicitly know P since it does not know b.

2. (Implicitly defining $e_k, \ldots, e_n$) Next, for every index i, t such that $k \le i \le n, 0 \le t \le k-1$ the simulator computes the Lagrange Coefficients

$$\lambda_t^i \leftarrow \prod_{m\in\{0,\ldots,k-1\}\setminus\{t\}} \frac{i-m}{t-m} \in \mathbf{Z}_q^*$$

satisfying that

$$e_i = P(i) = \sum_{t=0}^{k-1} \lambda_t^i P(t) \qquad (1)$$

[2]The index i is part of c_i to be able to explicitly reconstruct $\mathcal{I}$ and thus compute $\lambda_i^{\mathcal{I}}$ given a set C.

Notice that the simulator does not know the values e_i for $i = k, \ldots, n$. However, it can compute an implicit representation thereof by means of Equation 1. Namely $h_i = g_1^{e_i}$ can be computed as $h_i = (g_1^b)^{\lambda_0^i} \cdot h_1^{\lambda_1^i} \cdots h_{k-1}^{\lambda_{k-1}^i}$ for $i = k, \ldots, n$.

3. (Defining Γ) Let Γ be $g_2^{P(0)} = g_2^b$.

It can be seen that $e_1, \ldots, e_n$ follow the same distribution as in the real game and thus $\Pr[X_0] = \Pr[X_1]$.

Game 2 (Answering H queries) A list H^{list} with entries of the form $\mathbb{N} \times \mathcal{P} \times \mathbf{G}_1 \times \mathbf{Z}_q$ is created to answer adversarial queries to the random oracle H.

The simulator starts by choosing $j^\star \xleftarrow{\$} \{1, \ldots, q_H\}$. At each new query $p \in \mathcal{P}$ made by the adversary the simulator proceeds as follows:

1. If there exists an entry of the form $\langle \cdot, p, A, \cdot \rangle$ in H^{list} it answers $H(p) = A$.
2. Otherwise, let j be the greatest index appearing in H^{list}. We distinguish two cases:
   (a) $j = j^\star - 1$. In this case, the simulator adds $\langle j^\star, p, g_1^a, 1 \rangle$ to H^{list} and answers $H(p) = g_1^a$.
   (b) $j \neq j^\star - 1$. The simulator chooses $\lambda \xleftarrow{\$} \mathbf{Z}_q^*$, adds $\langle j+1, p, g_1^\lambda, \lambda \rangle$ to H^{list} and answers $H(p) = g_1^\lambda$.

This modification does not change the distribution of H's output and thus $\Pr[X_1] = \Pr[X_2]$.

Game 3 (Answering challenge ciphertexts set) Let p_0, p_1 be the plaintexts output by the adversary. Let $\langle j_0, p_0, A_0, \lambda_0 \rangle$ and $\langle j_1, p_1, A_1, \lambda_1 \rangle$ be the entries corresponding to plaintexts p_0, p_1 in H^{list}. The simulator proceeds as follows:

1. If $j_0, j_1 \neq j^\star$ the simulator aborts the game and outputs $\beta' \xleftarrow{\$} \{0,1\}$ as its guess to the BDDH problem.
2. Otherwise let $j_{\bar{\beta}}$ for $\bar{\beta} \in \{0,1\}$ be such that $H(p_{\bar{\beta}}) = g_1^a$. For $i \in \{r, \ldots, k-1\}$ the challenge ciphertext shares $c_i = \langle i, \eta_i, \gamma_i, \alpha_i \rangle$ are computed as

$$\eta_i = (g_1^a)^{e_i}, \quad \gamma_i = (g_2^c)^{r_i}, \quad \alpha_i = \mathbf{enc}(K_i, p)$$

with $K_i \leftarrow \mathbf{e}(g,g)^{abcr_i}$ for $r_i \xleftarrow{\$} \mathbf{Z}_q^*$.
Then $\Pr[X_3] = 2\Pr[X_2]/q_H$.

Game 4 (Answering non-challenge encryption queries) In this game we proceed to modify how encryption queries of the form $\mathsf{enc}(i, p)$, where $i \in \{r+1, \ldots, n\}$ are simulated. Let $\langle j, p, A, \lambda \rangle$ be the entry corresponding to plaintext p in H^{list}. Clearly $j \neq j^\star$, since the simulator did not abort at Game 3, and the IND game restricts $\mathsf{enc}(\cdot, \cdot)$ queries to plaintexts $p \neq p_0, p_1$. The simulator proceeds as follows: we have that $H(p) = g_1^\lambda$ for known $\lambda \in \mathbf{Z}_q^*$, and therefore $c_i = \langle i, \eta_i, \gamma_i, \alpha_i \rangle$, where

$$\eta_i = h_i^\lambda, \quad \gamma_i = g_2^{r_i}, \quad \alpha_i = \mathbf{enc}(K, p)$$

with $K \leftarrow \mathbf{e}(H(p), \Gamma)^{r_i}$ for $r_i \xleftarrow{\$} \mathbf{Z}_q^*$ is a correct encryption answer.

We conclude $\Pr[X_4] = \Pr[X_3]$.

Game 5 (Embedding BDDH challenge) The simulator sets the encryption keys to be random in the challenge ciphertext set. That is, for $i \in \{r, \ldots, k-1\}$ the component α_i in the challenge ciphertext shares $c_i = \langle i, \eta_i, \gamma_i, \alpha_i \rangle$ is changed to $\alpha_i = \mathbf{enc}(K_i, p)$, with $K_i \leftarrow T^{r_i}, r_i \xleftarrow{\$} \mathbf{Z}_q^*, T \xleftarrow{\$} \mathbf{G}_T$.

Clearly, an adversary distinguishing Game 4 and Game 5 implies a distinguisher for the BDDH problem, and thus we can claim there exists an adversary $\mathcal{B}_1$ such that $|\Pr[X_5] - \Pr[X_4]| \leq (k-1-r)\mathbf{Adv}^{\mathrm{BDDH}}_{\mathcal{IG},\mathcal{B}_1}(\ell) \leq (k-1)\mathbf{Adv}^{\mathrm{BDDH}}_{\mathcal{IG},\mathcal{B}_1}(\ell)$.

Game 6 (Embedding the DDH1 challenge) To ensure challenge ciphertexts do not leak unnecessary information the simulator proceeds to the final modification. Let p_β be such that $H(p_\beta) = g_1^a$. The simulator proceeds as follows. For $i \in \{r, \ldots, k-1\}$ the component η_i in the challenge ciphertext shares $c_i = \langle i, \eta_i, \gamma_i, \alpha_i \rangle$ is modified such that

$$\eta_i = (g_1^s)^{e_i}, \quad \gamma_i = (g_2^c)^{r_i}, \quad \alpha_i = \mathbf{enc}(K, p)$$

for $s \xleftarrow{\$} \mathbf{Z}_q^*$. With this change we ensure that the components $\eta_i, \gamma_i, \alpha_i$ are independently distributed (recall that $K_i \xleftarrow{\$} \mathbf{G}_T$ and $r_i \xleftarrow{\$} \mathbf{Z}_q^*$). An adversary $\mathcal{A}$ distinguishing between Game 5 and Game 6 implies a distinguisher for the DDH1 problem. We can conclude then the existence of an adversary $\mathcal{B}_2$ such that $|\Pr[X_6] - \Pr[X_5]| \leq (k-1-r)\mathbf{Adv}^{\mathrm{DDH1}}_{\mathcal{IG},\mathcal{B}_2}(\ell) \leq (k-1)\mathbf{Adv}^{\mathrm{DDH1}}_{\mathcal{IG},\mathcal{B}_2}(\ell)$.

Now, since in every ciphertext share c_i the corresponding components are independently distributed from each other, we conclude there exists an adversary $\mathcal{B}_3$ such that $|\Pr[X_6] - 1/2| = \mathbf{Adv}^{\mathrm{ae-ot}}_{\mathrm{AE},\mathcal{B}_3}(1^\ell)$.

Finally, collecting all intermediate probabilities we obtain the targeted result. □

4. DISTRIBUTED ENCRYPTION WITH FORWARD SECURITY

Plain distributed encryption does not fulfil the *forward privacy* property inherent to revocable privacy, namely, that no redesign of the system should allow the release of previously gathered information if the latter falls outside the data release rule (in our case that less than a threshold of k ciphertext shares are in possession of the combiner). A typical scenario is found when the relevant authorities, that in principle set up a data collection system with privacy-preserving guarantees, change their mind and seize senders to reveal their secret encryption keys. In this case secrecy in plain distributed encryption is not guaranteed anymore, since indistinguishability of ciphertext shares holds when the combiner (e.g. the authority) knows up to $k-1$ secret encryption keys.

Our approach to mitigate the damage caused by a potential system redesign is to use a *key-evolving distributed encryption* scheme. In a key-evolving scheme, the operation time of the scheme is divided into stages, and at each new stage the encryption keys are changed, while keys corresponding to the previous period are deleted. Such a key-evolving distributed encryption scheme is said to have *forward security* if, when the adversary breaks in and learns the secret keys corresponding to the current stage, then this adversary can not abuse any data collected in the past.

4.1 Definition

Formally, a k-out-of-n *key-evolving distributed encryption* scheme with lifetime divided into s stages consists of four algorithms KDE = (**Gen**, **UpdKey**, **Enc**, **Comb**):

Gen$(1^\ell, k, n, s)$ Given a security parameter 1^ℓ, number of users n, threshold parameter k and life span consisting on s stages, this function generates initial encryption keys $E_{1,1}, \ldots, E_{1,n}$ belonging respectively to senders $S_1, \ldots, S_n$. Additionally it outputs the description of plaintext and ciphertext spaces $\mathcal{P}$ and $\mathcal{C}$. A plaintext p is called admissible if $p \in \mathcal{P}$; admissible ciphertexts are defined analogously.

UpdKey$(E_{\sigma-1,i})$ The key $E_{\sigma,i}$ at any stage σ is obtained from the key $E_{\sigma-1,i}$ at the previous stage. After the key $E_{\sigma,i}$ has been built, the key $E_{\sigma-1,i}$ is deleted. Aborts if the current stage equals s.

Enc$(E_{\sigma,i}, p)$ Given an encryption key $E_{\sigma,i}$ corresponding to sender i at stage σ and a plaintext p, this function returns a ciphertext share $c_{\sigma,i}$.

Comb(C) Given a set $C = \{c_{\sigma_1,i_1}, \ldots, c_{\sigma_k,i_k}\}$ consisting of k ciphertext shares, **Comb**(C) either returns a plaintext p or error.

Every key-evolving distributed encryption scheme must satisfy the following correctness requirement.

Correctness Let $E_{\sigma,1}, \ldots, E_{\sigma,n}$ be the encryption keys at any stage $\sigma \in \{1, \ldots, s\}$, obtained by successively running the corresponding key generation and update algorithms. Let $C = \{c_{\sigma_1,i_1}, \ldots, c_{\sigma_k,i_k}\}$, where $c_{\sigma_t,i_t} = \mathbf{Enc}(E_{\sigma_t,i_t}, p_t)$ for some sender i_t, stage σ_t and plaintext p_t. Then **Comb**(C) returns p with overwhelming probability if and only if $p_1 = \ldots = p_k = p$, as well as $\sigma_1 = \ldots = \sigma_k = \sigma$, while $i_1, \ldots, i_k$ are pairwise different (and returns error otherwise).

4.2 Security definition

Roughly speaking, the security of a key-evolving distributed encryption scheme is defined as follows: a combiner who at stage σ breaks in and obtains the whole collection of current secret encryption keys $E_{\sigma,1}, \ldots, E_{\sigma,n}$, should learn (almost) no information on any plaintext p contained in a set of $k - r - 1$ ciphertext shares, such that these ciphertext shares were produced at a stage $\sigma^\star < \sigma$ in which the combiner had only corrupted up to r senders.

We now describe the formal security definition for KDE schemes, called *forward security*, and referred to as FSIND. It is obtained by extending the IND security notion to the key-evolving setting.

Definition 4.1 (FSIND) *Let us consider a* (k, n, s)-KDE = (**Gen**, **UpdKey**, **Enc**, **Comb**) *key-evolving distributed encryption scheme. Let us define the following game between a challenger and an adversary* $\mathcal{A}$*:*

Phase 0 (Only for static adversaries) *The adversary outputs a set of* r *pairs* $I_c = \{(\sigma_1, i_1), \ldots, (\sigma_r, i_r)\}$*, where* (σ_t, i_t) *are such that* $\sigma_t \in \{1, \ldots, s\}$ *and* $i_t \in \{1, \ldots, n\}$ *and* $0 \leq r < k$*. A pair* $(\sigma_t, i_t) \in I_c$ *denotes that the adversary corrupts sender* S_{i_t} *at stage* σ_t*. In this case the adversary receives* E_{σ_t,i_t} *as soon as* σ_t *is entered. It also outputs a stage number* $\sigma^\star$ *and* $k - 1 - r$ *pairwise different indexes* $I_{nc} = \{i_{r+1}, \ldots, i_{k-1}\}$*, that respectively correspond to the stage number and senders on which the adversary will be challenged.*

Setup *The challenger runs* $(E_{1,1}, \ldots, E_{1,n}) \leftarrow \mathbf{Gen}(1^\ell, k, n, s)$.

Find *The set of queried plaintexts* $Q_{\sigma'}$ *is initialised to* $\varnothing$ *for all stages* $\sigma' \in \{1, \ldots, s\}$*.. For adaptive adversaries the set* I_c *of corrupted senders is initialized to* $I_c = \varnothing$*. The current stage number* σ *is initialized to* $\sigma \leftarrow 1$*.*

The adversary can issue three types of queries:

- *On encryption queries of the form* $\mathsf{enc}(i, p)$*, where* $i \in \{1, \ldots, n\}$*,* $(t, i) \notin I_c$ *for any* $t \leq \sigma$*, and* p *is an admissible plaintext, the adversary receives* $\mathbf{Enc}(E_{\sigma,i}, p)$*, where* σ *is the current stage;* p *is added to* Q_σ*.*
- *On next-stage queries* $\mathsf{next}()$*, the challenger updates the current stage as* $\sigma \leftarrow \sigma + 1$ *and updates the encryption keys as* $E_{\sigma,i} \leftarrow \mathbf{UpdKey}(E_{\sigma-1,i})$ *for* $i \in \{1, \ldots, n\}$*.*
- (Only for adaptive adversaries) *On queries of the form* $\mathsf{corrupt}(I_{tbc})$*, where* $I_{tbc} \subset \{1, \ldots, n\}$ *is a possibly non-empty set, the challenger proceeds as follows.*
  - *for all* $i \in I_{tbc}$*, it adds* (σ, i)*, where* σ *is the current stage, to* I_c*. If* $|I_c| > k - 1$ *then the game aborts and the adversary looses.*
  - *for all pairs* $(\sigma, i) \in I_c$ *where* σ *is the current stage, the adversary receives* $E_{\sigma,i}$*.*

The cardinality of I_c *at the end of the* Find *phase is denoted by* r*, and it holds that that* $r \leq k - 1$*.*

Challenge *In the case of adaptive adversaries,* $\mathcal{A}$ *outputs a challenge stage number* $\sigma^\star < \sigma$ *and* $k - 1 - r$ *indexes* $I_{nc} = \{i_{r+1}, \ldots, i_{k-1}\}$ *corresponding to the senders on which* $\mathcal{A}$ *wants to be challenged.*

$\mathcal{A}$ *outputs two equal-length plaintexts* $p_0, p_1 \in \mathcal{P}$ *where* $p_i \notin Q_{\sigma^\star}$*. For each* $i \in I_{nc}$ *we require that* $(t, i) \notin I_c$ *for all* $t \in \{1, \ldots, \sigma^\star\}$*. Finally the challenger chooses* $\beta \xleftarrow{\$} \{0, 1\}$ *and returns*

$$\{c_{\sigma*,i} \mid i \in I_{nc}\} \text{ and } \{E_{\sigma,1}, \ldots, E_{\sigma,n}\} \ ,$$

where $c_{\sigma*,i} = \mathbf{Enc}(E_{\sigma*,i}, p_\beta)$ *for* $i \in I_{nc}$*.*

Guess *The adversary* $\mathcal{A}$ *outputs a guess* $\beta' \in \{0, 1\}$*. The adversary wins the game if* $\beta = \beta'$*.*

Define $\mathcal{A}$*'s advantage as* $\mathbf{Adv}^{\text{FSIND}}_{\text{KDE},\mathcal{A}}(1^\ell) = \left|\Pr[\beta' = \beta] - 1/2\right|$*. A scheme* KDE *is called* forward-secure (FSIND secure) *if* $\mathbf{Adv}^{\text{FSIND}}_{\text{KDE},\mathcal{A}}(1^\ell)$ *is negligible for every PPT adversary* $\mathcal{A}$*.*

4.3 Construction

A FSIND secure KDE scheme can be obtained by applying a generic transformation that consists of building s independent copies of the IND secure DE scheme from Section 3.2. The resulting scheme is described below. We omit here the statement of its security theorem due to the lack of space. We briefly mention that security is based on the

same assumptions as the plain DE scheme; the tightness of the security reduction is degrading linearly on the number of stages s.

At each stage σ each user i uses a separate key, which is stored as the head of a list of keys $E_{\sigma,i}$. Initially user i owns the list $E_{1,i}$ of s keys. When progressing to the next stage, the head of this list is removed. The list functions **head**(), **tail**() and **cons**() are defined as usual. () denotes the empty list.

Gen$(1^\ell, k, n, s)$ First generate an asymmetric pairing, by running

$$\langle \mathbf{e}(\cdot,\cdot), \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_T, g_1, g_2, q\rangle \leftarrow \mathcal{IG}(1^\ell).$$

For all i, let $E_{s+1,i} = ()$, the empty list. Recursively define $E_{\sigma,i}$ for σ from s to 1 as follows. For each stage, generate n fresh secret keys $(E_1, \ldots, E_n) :=$ DE.**Gen**$(1^\ell, k, n)$. For all i, set $E_{\sigma,i} =$ **cons**$(E_i, E_{\sigma+1,i})$. Note that although the pairing function remains constant, the hash function H used in each of the stages is different. The space of plaintexts $\mathcal{P}$ is set to be that of AE. Return $(E_{1,1}, \ldots, E_{1,n})$.

UpdKey$(E_{\sigma-1,i})$ Set $E_{\sigma,i} :=$ **tail**$(E_{\sigma-1,i})$. After the key $E_{\sigma,i}$ has been built, the key $E_{\sigma-1,i}$ is erased.

Enc$(E_{\sigma,i}, p)$ Return DE.**Enc**(**head**$(E_{\sigma,i}), p)$.

Comb(C) Return DE.**Comb**(C).

5. APPLYING DISTRIBUTED ENCRYPTION

Let us return to the motivating problems described in Section 1.1, and see how distributed encryption can help solve these problems.

To solve the database logging problem in a privacy friendly way, one can use distributed encryption as follows. Each of the n databases acts as a sender, and each database has its own distributed encryption key $E_{1,i}$. Initial keys are generated using **Gen**$(1^\ell, k, n, s)$, for a suitably chosen threshold k. To create a log entry for a particular transaction, database i encrypts the identity of the user u using his own encryption key and stores the resulting ciphertext share **Enc**$(E_{\sigma,i}, u)$ in the transaction log for the record involved. Let $L_i(r)$ be the transaction log for record r in database i. In order to determine who was involved in transactions concerning records $r_{i_1}, \ldots, r_{i_k}$ from k different databases $i_1, \ldots, i_k$, the combiner tries all possible combinations of ciphertext shares from the k logging sets $L_{i_1}(r_{i_1}), \ldots, L_{i_k}(r_{i_k})$. With at most ℓ entries in each of these sets, the total time complexity of this operation equals ℓ^k. If particular records are only accessed by relatively few users, this complexity is manageable.

To solve the canvas cutter problem in a privacy friendly way, one could choose to retain the data coming from a single ANPR system for only a couple of hours. But this is only a procedural measure, which is not good enough (as discussed in section 1.2). A straightforward application of the distributed encryption scheme KDE introduced above however allows us to solve the canvas cutters more thoroughly.

Each of the n ANPR systems i acts as a sender, using the distributed encryption scheme with its own key. Initial keys are generated using **Gen**$(1^\ell, k, n, s)$. Keys are updated in the obvious manner at the start of a new stage. Each number plate p recorded by an ANPR i is encrypted immediately (and not stored locally) and sent[3] as ciphertext share **Enc**$(E_{\sigma,i}, p)$ to a central server using a secure channel that guarantees authenticity and confidentiality of the messages[4].

The server stores, for each stage σ and for each ANPR node i, the list $C_{i,\sigma}$ of ciphertext shares sent by ANPR node i during stage σ. In order to find a set of k encryptions of the same number plate during stage σ, all combinations of taking k elements, one from each list $C_{i,\sigma}$, need to be tried. Each of the possible combinations C is passed to **Comb**(C). Let each list contain m entries. Then the associated time complexity becomes $m^k \binom{n}{k}$ (because we can choose k-out-of-n different list combinations, and within each combination of k lists we can take m^k different samples).

Clearly, this is a highly inefficient solution. We are currently investigating different approaches to reduce the time complexity considerably.

6. CONCLUSIONS

We have shown that techniques to implement revocable privacy exist, and introduced non-interactive (forward secure) distributed encryption as a new element of this toolbox.

However, these techniques are only applicable in specific cases and henceforth more general techniques need to be developed. These are investigated in the Revocable Privacy project, of which the present paper is a first result. Progress will be reported through the website `www.revocable-privacy.org`. We note that the underlying design principles to achieve revocable privacy can already be used, using either general trusted third parties techniques or special purpose mechanisms. Another purpose of our work is to raise a more general awareness of this possibility.

We leave as an open problem the construction of a non-interactive distributed encryption scheme not resorting to the random oracle heuristic or not based on identity-based encryption.

Also, more efficient solutions to the canvas cutters problem are needed. This may be possible by moving to a richer setting. For example, if two way communication between the car and the ANPR system is allowed (for instance using a system embedded in the number plate of the car), then k-times anonymous authentication [11] may be applied. Each day, a car receives one such credential, and when visiting a parking place the ANPR requests an anonymous authentication using that credential. This solves a slightly different problem, because if one uses n-times anonymous authentication then the number plate of a car that visits the *same* parking space n times will be revealed (whereas for distributed encryption this car should have visited n *different* parking spaces). Moreover, immediate applicability of such a system is limited, since widespread embedding of cryptography in the cars seems currently unrealistic. Further research is necessary to explore these issues.

[3]Of course, this could also be implemented in store-and-forward kind of setup, where the ANPR nodes do store the encrypted shares for a while until they are being collected to be forwarded to the central server.

[4]Note that the definition of a distributed encryption scheme allows anyone to act as a combiner. Confidentiality of the messages is therefore required to ensure that only the server can act as combiner.

7. REFERENCES

[1] Masayuki Abe and Miyako Ohkubo. Provably secure fair blind signatures with tight revocation. In Colin Boyd, editor, *ASIACRYPT*, LNCS 2248, pages 583–602. Springer, 2001.

[2] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic -taa. In Roberto De Prisco and Moti Yung, editors, *SCN*, LNCS 4116, pages 111–125. Springer, 2006.

[3] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2004.

[4] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[5] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[6] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2006.

[7] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[8] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 237–254. Springer, 2010.

[9] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptology*, 18(3):219–246, 2005.

[10] Jan Camenisch, Thomas Groß, and Thomas S. Heydt-Benjamin. Rethinking accountable privacy supporting services: extended abstract. In Elisa Bertino and Kenji Takahashi, editors, *Digital Identity Management*, pages 1–8. ACM, 2008.

[11] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

[12] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *SCN*, LNCS 4116, pages 141–155. Springer, 2006.

[13] David Chaum. Untraceable electronic mail, return adresses, and digital pseudonyms. *Comm. ACM*, 24(2):84–88, 1981.

[14] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO*, LNCS 403, pages 319–327. Springer, 1988.

[15] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *FOCS*, pages 383–395. IEEE, 1985.

[16] Cécile Delerablée and David Pointcheval. Dynamic threshold public-key encryption. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 317–334. Springer, 2008.

[17] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[18] Augusto Jun Devegili, Michael Scott, and Ricardo Dahab. Implementing cryptographic pairings over barreto-naehrig curves. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 197–207. Springer, 2007.

[19] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2003.

[20] C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A family of implementation-friendly bn elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.

[21] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[22] Jaap-Henk Hoepman. Revocable privacy. *ENISA Quarterly Review*, 5(2):16–17, June 2009.

[23] Lawrence Lessig. *Code and other laws of cyberspace*. Basic Books, 1999.

[24] Keith M. Martin, Reihaneh Safavi-Naini, Huaxiong Wang, and Peter R. Wild. Distributing the encryption and decryption of a block cipher. *Des. Codes Cryptography*, 36(3):263–287, 2005.

[25] Bruce Schneier. What our top spy doesn't get: Security and privacy aren't opposites. *Wired*, January 2008.

[26] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[27] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT*, pages 1–16, 1998.

[28] Nigel P. Smart and Frederik Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007.

[29] Daniel J. Solove. *Understanding Privacy*. Harvard University Press, 2008.

[30] Markus Stadler. *Cryptographic Protocols for Revocable Privacy*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 1996.

[31] Markus Stadler, Jean-Marc Piveteau, and Jan

Camenisch. Fair blind signatures. In *EUROCRYPT*, pages 209–219, 1995.

[32] Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2006.

[33] Eric R. Verheul and Henk C. A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In *EUROCRYPT*, pages 119–133, 1997.

[34] Samuel D. Warren and Louis D. Brandeis. The right to privacy [The implicit Made Explicit]. *Harvard Law Review*, IV(5):193–220, December 15 1890.

[35] Hoeteck Wee. Threshold and revocation cryptosystems via extractable hash proofs. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 589–609. Springer, 2011.