Privacy-Preserving Plaintext-Equality of Low-Entropy Inputs

Sébastien Canard[1], David Pointcheval[2,3], Quentin Santos[1,2,3(✉)], and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
quentin.santos@orange.com
[2] DIENS, CNRS École normale supérieure, PSL University, Paris, France
[3] INRIA, Paris, France

Abstract. Confidentiality requires to keep information away from the eyes of non-legitimate users, while practicality necessitates to make information usable for authorized users. The former issue is addressed with cryptography, and encryption schemes. The combination of both has been shown to be possible with advanced techniques that permit to perform computations on encrypted data. Searchable encryption concentrates on the problem of extracting specific information from a ciphertext.

In this paper, we focus on a concrete use-case where sensitive tokens (medical records) allow third parties to find matching properties (compatible organ donor) without revealing more information than necessary (contact information).

We reduce such case to the plaintext-equality problem. But in our particular application, the message-space is of limited size or, equivalently, the entropy of the plaintexts is small: public-key existing solutions are not fully satisfactory. We then propose a suitable security model, and give an instantiation with an appropriate security analysis.

1 Introduction

With the advance of computing and networking, cryptography has evolved from providing straightforward guarantees such as confidentiality, integrity and authenticity to providing many complex features. In particular, much research has been done on the task of performing various kinds of operations on encrypted data. The most well-known topics include fully homomorphic encryption and garbled circuits, whose practical realization would bring into the realm of possibility many applications that would have deemed as magical and unlikely a few decades ago.

A simpler but still useful problem is that of extracting information from a ciphertext. This can include allowing the testing of a single bit of a ciphertext, testing whether a ciphertext contains a particular value or not, whether it includes this value as a substring or not. Searchable encryption already allows

B. Preneel and F. Vercauteren (Eds.): ACNS 2018, LNCS 10892, pp. 262–279, 2018.
https://doi.org/10.1007/978-3-319-93387-0_14

many practical uses, such as spam filtering or threat detection in encrypted traffic, but has the advantage over more generic cryptocomputing techniques to have much more efficient instantiations.

1.1 Motivation: Organ Donation

In this paper, we take the specific use case of organ donation as a motivation and derive our constraints from issues related to the actual realization of a solution. With this approach, we devise a method to solve the problem using a new kind of cryptographic primitive. This work is connected to the Kidner Project[1] which aims at providing a solution for kidney donation matching using a blockchain.

Organ Donation. Organ transplant requires the donor and the recipient to be compatible, so as to reduce the risks of a graft rejection. In practice, this means that they must be of similar age, have compatible antigens (blood type and human leukocyte antigen (HLA) system), etc. This also means that a list of donors and a list of recipients must be maintained: when a new donor (respectively recipient) enters the system, the new entry should be checked against existing recipients (respectively donors).

Donors are usually willing to give an organ only to a close relative; this means that the pool for potential matches is very restricted. To lift this constraint, the common approach is to pair a willing donor with a recipient, disregarding medical incompatibility. Then, the problem consists in finding two pairs that are mutually compatible (the donor of a pair is compatible with the recipient of the other pair), so that each donor accepts to give their organ to the other recipient.

To enlarge the pool of potential matches further, we can consider the directed graph over donor-recipient pairs, and draw an edge from a donor to a compatible recipient. The general problem is that of finding simple cycles in this graph; the simplest version is a cycle of length 2, involving two donor-recipient pairs, but longer cycles allow more compatibility solutions.

Confidential information on donors and recipients is usually managed by non-profit entities (e.g., Organ Procurement Organizations) who exchange information in order to find matching pairs. Optimizing the number of matches requires merging information from as many sources as possible, but threatens the confidentiality of patient records.

In this paper, we explore a method to encrypt such records in order to obtain both data confidentiality and the ability to test for compatibility.

Compatibility Matching by Equality Check. We show how to reduce this problem to that of testing for equality.

Blood Type. First, consider compatibility on blood type alone. Donor and recipient can each be O, A, B or AB. The compatibilities are shown on Fig. 1. The recipient will generate a record for each of the compatible donors: for instance,

[1] https://www.kidner-project.com.

a recipient of blood type A will generate a record whose field `Blood Type` is set to O, and another where this field is set to A.

		Donor			
		O	A	B	AB
Recipient	O	✓			
	A	✓	✓		
	B	✓		✓	
	AB	✓	✓	✓	✓

Fig. 1. Blood compatibility

Remark 1. Records should not be linkable one to another; however, if they are, the number of compatible blood types can be hidden by padding with filler values. This can be done by constant incompatible values between donors and receivers.

Age Group. Second, consider compatibility on age alone. This criterion is soft: individuals of similar ages should match, but we do not want to discriminate them into separate age groups. Instead, we use overlapping age groups by letting the recipient list acceptable age groups for the donor. For instance, a 19-year-old recipient would list age groups $[12-19]$ and $[20-39]$.

Human Leukocyte Antigens (HLA). Each individual is associated with six variables HLA-{A,B,C,E,F,G}. Two individuals are considered to be HLA-wise compatible when at least three of these variables match. This time, the recipient and the donor each generate a record for each of the 20 combinations of three HLAs (binomial of 3 out of 6).

All Together. By combining these brute-force-inspired solutions, we expect an individual to generate on average less than 200 records. The overhead is non-negligible but overcomes complex matching procedures.

We now consider an encrypted version of an individual's record, called a "fingerprint", and we want to test whether two fingerprints are equal or not.

1.2 Related Work

Testing whether two ciphertexts hold the same value can be easily done in some contexts. For instance, when all the relevant ciphertexts are encrypted under the same key, and when a deterministic encryption scheme is used, it is sufficient to compare the two outputs [BBO06]. It is also possible to allow testing a ciphertext against a plaintext value by simply using deterministic public key encryption or a one-way function (optionally along with a classical encryption of the message, if decryption is needed).

Public Key Encryption with Equality Test (PKEET) allows testing plaintext-equality between two ciphertexts [YTHW10], while encryption can be done by anyone. One may think of deterministic public key encryption schemes as a subset of PKEET schemes.

In other contexts, more elaborate schemes are needed. In searchable encryption [SWP00, BDOP04], each word w from the input message m is encrypted separately as $s = \mathsf{PEKS}(w)$. The PEKS scheme then allows other participants to search for a keyword w' in m simply by testing whether $w = w'$ given s, as well as a trapdoor value $T_{w'}$. Another variant allows testing between a ciphertext and a plaintext, without a trapdoor [CFGL12]. In different settings, it is possible to use interactive protocols, such as for private matching and set intersection [FNP04].

1.3 Our Contribution

Fingerprinting and Testing Keys. It is important to note that PKEETs rely on the high min-entropy of the message distribution [LZL13]. Indeed, an attacker may test a target ciphertext against arbitrary messages to conduct a brute-force search over the message space, since encryption is public. We thus have to exclude this approach.

We introduce the notion of fingerprint, a kind of probabilistic ciphertext that allows plaintext-equality testing. Private fingerprinting (generation of fingerprint) allows us to provide semantic security [GM84] (a.k.a. indistinguishability or polynomial security). Alternatively (or additionally), it is possible to make the testing private. We consider all scenarios in our generic model but, since legitimate users need to run many more plaintext-equality testings than fingerprintings (contrary to an adversary), we are interested in a mechanism where testing is public and fingerprinting is private.

Finally, we would prefer non-interactive matching protocols: in our motivation, searching for a cycle requires many compatibility tests; using an interactive protocol would incur an important communication cost and make the system less robust to network hazards.

Blind and Threshold Fingerprinting. We could entrust the fingerprinting key to a trusted third party (TTP) to control the number of queries but we want the fingerprints to be generated without seeing the input messages. Thus, we will use blind fingerprinting, which is similar to blind signing [Cha82], but we do not require unlinkability.

This is not enough for privacy: the fingerprinter can still generate and test for plaintext-equality as many fingerprints as they want (assuming public-key testing). To actually achieve a decent level of privacy, we therefore split them into several fingerprinters: without collusions above some threshold, no information is leaked about the input message (blindness) and no brute-force attack is possible.

1.4 Organization

In Sect. 2, we first draw a generic model for fingerprinting under the constraints listed above; in particular, we take into consideration both public and private fingerprinting and both public and private testing. Then, in Sect. 3, we introduce the two assumptions which our construction relies on, one of which is new to this paper, and holds in the generic bilinear group model. Finally, we propose a construction for our new scheme in Sect. 4, show its security, and present the blind and threshold variants which extend the privacy of the user.

The proof for the new assumption in the generic bilinear group model is postponed to Appendix A, and the proofs of security of our construction can be found in Subsect. 4.3.

2 Fingerprinting Scheme

In this section, we first define a more general fingerprinting mechanism, where the generation of fingerprints and the testing algorithm require keys that can be either private or public. We will later focus on our concrete scenario, with private fingerprint generation and public testing.

2.1 Description

We consider three kinds of players:

- the **fingerprinter** who generates the fingerprints of messages using the fingerprinting key. We consider this operation in the honest-but-curious framework, since we eventually split the fingerprinter into several parties, each holding a share of the fingerprinting key;
- the **tester** who checks whether two fingerprints correspond to the same message or not, using the testing key;
- the **users** who have access to the list of fingerprints, and who may query for new fingerprints (through the fingerprinter) and compare fingerprints (through the tester).

We stress however that the fingerprinting and testing keys may be either public or private. When a key is secret, the users have to interact with the owner of the key to benefit from the corresponding functionality; when it is public, the users can act on behalf of the fingerprinter or the tester. The choice of publishing a key or keeping it private will depend on the scenario under consideration.

Finally, we eventually choose to take advantage of the asymmetric nature of our use case: bipartite matching, between people from two different groups (donors and receivers). So, we will manipulate two kinds of fingerprints: "left" and "right" fingerprints in this generic specification.

We thus define four protocols:

- $\mathsf{KeyGen}(1^k)$ creates the global parameters and the left and right fingerprinting keys lk and rk as well as the testing key tk, for security parameter k;

- $\mathsf{LFingerprint}(\mathsf{lk}, m)$, given a left-fingerprinting key lk and a message m, outputs a left-fingerprint f_L;
- $\mathsf{RFingerprint}(\mathsf{rk}, m)$, given a right-fingerprinting key rk and a message m, outputs a right-fingerprint f_R;
- $\mathsf{Test}(\mathsf{tk}, f_L, f_R)$, given a testing key tk, a left-fingerprint f_L and a right-fingerprint f_R, reveals whether they correspond to the same message or not.

As already noted above, these procedures can be either private or public, and they can be algorithms to be run offline, or interactive protocols. Various situations can be envisioned according to the secrecy of the fingerprinting and testing keys.

- Testing and fingerprinting keys public: security solely rely on the high entropy of the inputs (message-locked encryption, as in PKEETs);
- Fingerprinting keys private only: our use case, where we want to limit the generation of fingerprints, but allow anyone to test freely for compatibility;
- Testing key private only: this can be relevant if the message space is very constrained, when even a few tests could leak too much information;
- Testing and fingerprinting keys private: this has the highest security guarantee, but is usually impractical unless performing very few queries is enough.

Remark 2. We can choose to have one of the fingerprinting keys private, and the other public. This setup can give some flexibility for specific use cases.

2.2 Security Model

Let us now make more precise the security notions we want to achieve. Since secret information can include the fingerprinting keys lk and rk, the testing key tk, and the users' input messages, we consider the following security properties:

1. unforgeability of fingerprinting (even against the tester[2]);
2. one-more indistinguishability of testing (even against the fingerprinter[3]);
3. privacy of the user w.r.t. the tester;
4. privacy of the user w.r.t. the fingerprinter.

Authentication of the Fingerprinter. The *raison d'être* of the fingerprinter is to generate fingerprints, so unforgeability guarantees that no one else can do so: even a collusion between the tester (access to the testing key) and users (queries to the fingerprinter) should be unable to generate a valid fingerprint that was not provided by the fingerprinter. This implies that the fingerprinting key is not leaked during this game. We formally define Fingerprint-Unforgeability ($\mathsf{FP{-}UF}$).

[2] Even the testing key should give no advantage to anybody in generating fingerprints.
[3] Even the fingerprinting key should give no advantage to anybody in making tests.

Definition 1 (FP–UF). *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{LFingerprint}, \mathsf{RFingerprint}, \mathsf{Test})$ *be the scheme presented above, and let* $\mathcal{A}$ *be a polynomial-time adversary. Let*

$$\mathsf{Adv}^{\mathsf{FP-UF}}_{\Pi,L}(\mathcal{A}) = \Pr\left(\begin{array}{l}(\mathsf{lk},\mathsf{rk},\mathsf{tk}) \xleftarrow{\$} \mathsf{KeyGen}(1^k), (m^\star, f_L^\star) \leftarrow \mathcal{A}^{\mathcal{L}}(\mathsf{rk},\mathsf{tk}),\\ f_R \leftarrow \mathsf{RFingerprint}(\mathsf{rk}, m^\star) : \mathsf{Test}(\mathsf{tk}, f_L^\star, f_R) = 1\end{array}\right)$$

where $\mathcal{L}$ *refers to the left-fingerprinting oracle, which answers to queries on message* m_i *with* $f_{L,i} = \mathsf{LFingerprint}(\mathsf{lk}, m_i)$*. We insist that* $m^\star$ *is distinct from any queried* m_i*.*

We similarly define $\mathsf{Adv}^{\mathsf{FP-UF}}_{\Pi,R}$*, with the left-fingerprinting key but access to the right-fingerprinting oracle. We say that* Π *is* $(t,\varepsilon)-\mathsf{FP{-}UF}$*-secure when both* $\mathsf{Adv}^{\mathsf{FP-UF}}_{\Pi,L}(\mathcal{A}) \leq \varepsilon$ *and* $\mathsf{Adv}^{\mathsf{FP-UF}}_{\Pi,R}(\mathcal{A}) \leq \varepsilon$ *for any* $\mathcal{A}$ *running within time* t*.*

Authentication of the Tester. The purpose of the tester is to help the user to test plaintext equality between fingerprints. But even a collision between the fingerprinter (access to the fingerprinting key) and users (queries to the tester), should be unable to guess the result of another test. This implies that the testing key is not leaked. We formally define Testing-Indistinguishability (T−IND).

Definition 2 (T–IND). *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{LFingerprint}, \mathsf{RFingerprint}, \mathsf{Test})$ *be the scheme presented above, and* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *a polynomial-time adversary. Let*

$$\mathsf{Adv}^{\mathsf{T-IND}}_{\Pi,L}(\mathcal{A}) = \left|\Pr\left(\begin{array}{l}(\mathsf{lk},\mathsf{rk},\mathsf{tk}) \xleftarrow{\$} \mathsf{KeyGen}(1^k),\\ (m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{T}}(\mathsf{lk},\mathsf{rk}), f_L \leftarrow \mathsf{LFingerprint}(\mathsf{lk}, m_0),\\ b \xleftarrow{\$} \{0,1\}, f_R \leftarrow \mathsf{RFingerprint}(\mathsf{rk}, m_b),\\ b' \leftarrow \mathcal{A}_2^{\mathcal{T}}(s, f_L, f_R) : b' = b\end{array}\right) - \frac{1}{2}\right|$$

where $\mathcal{T}$ *refers to the testing oracle, who answers to queries on fingerprints* f_L, f_R *with* $\mathcal{T}(f_L, f_R) = \mathsf{Test}(\mathsf{tk}, f_L, f_R)$*. We require that the attacker does not submit the challenge fingerprint* f_R *to the testing-oracle.*

We define $\mathsf{Adv}^{\mathsf{T-IND}}_{\Pi,R}(\mathcal{A})$ *in a similar fashion. We say that* Π *is* $(t,\varepsilon)-\mathsf{T{-}IND}$*-secure if both* $\mathsf{Adv}^{\mathsf{T-IND}}_{\Pi,L}(\mathcal{A}) \leq \varepsilon$ *and* $\mathsf{Adv}^{\mathsf{T-IND}}_{\Pi,R}(\mathcal{A}) \leq \varepsilon$ *for any adversary* $\mathcal{A}$ *running within time* t*.*

One can note that for such a strong notion of indistinguishability, which only excludes the challenge fingerprints from being queried to the testing-oracle, the fingerprints must be non-malleable.

Privacy of the User. This security notion adapts semantic security to our scheme: given access to the even a collusion between the tester (access to the testing key) and users (queries to the fingerprinter) should not be able to distinguish a fingerprint of a message m_0 from a fingerprint of a message m_1 (unless they know a fingerprint of m_0 or of m_1). Furthermore, the collusion could include one of the two fingerprinting keys (but not both): give the left-fingerprinting key when proving the semantic security of left-fingerprinting, and the right-fingerprinting key when proving the semantic security of right-fingerprinting. We formally define Fingerprint-Indistinguishability (FP−IND).

Definition 3 (FP–IND). *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{LFingerprint}, \mathsf{RFingerprint}, \mathsf{Test})$ *be the scheme presented above, and let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a polynomial-time adversary. Let*

$$\mathsf{Adv}^{\mathsf{FP-IND}}_{\Pi,L}(\mathcal{A}) = \left| \Pr \left(\begin{array}{l} (\mathsf{lk}, \mathsf{rk}, \mathsf{tk}) \xleftarrow{\$} \mathsf{KeyGen}(1^k), (m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{R}}(\mathsf{lk}, \mathsf{tk}), \\ b \xleftarrow{\$} \{0,1\}, f_L \leftarrow \mathsf{LFingerprint}(\mathsf{lk}, m_b), \\ b' \leftarrow \mathcal{A}_2^{\mathcal{R}}(s, f_L) : b' = b \end{array} \right) - \frac{1}{2} \right|$$

where $\mathcal{R}$ *refers to the right-fingerprinting oracle, which answers to queries on message* m'_i *with* $\mathcal{R}(m'_i) = \mathsf{RFingerprint}(\mathsf{rk}, m'_i)$*. We insist that* $m'_i \notin \{m_0, m_1\}$ *for any queries to* $\mathcal{R}$*.*

We define $\mathsf{Adv}^{\mathsf{FP-IND}}_{\Pi,R}(\mathcal{A})$ *similarly. We say that* Π *is* $(t, \varepsilon) - \mathsf{FP-IND}$*-secure if both* $\mathsf{Adv}^{\mathsf{FP-IND}}_{\Pi,L}(\mathcal{A}) \leq \varepsilon$ *and* $\mathsf{Adv}^{\mathsf{FP-IND}}_{\Pi,R}(\mathcal{A}) \leq \varepsilon$ *for any adversary* $\mathcal{A}$ *running within time* t*.*

Note that fingerprinting generation itself should not reveal anything about the message that is being fingerprinted: the view of the fingerprinter should be the same regardless of the message. Like in blind signatures [Cha82], no adversary playing the role of fingerprinter should be able to distinguish a fingerprinting of m_0 from a fingerprinting of m_1. However, if the fingerprinter sees the resulting fingerprint and locally generates a fingerprint for m_0, they could easily distinguish between the two cases. To avoid this, the fingerprinter should be split into several parties that need to cooperate to create a new fingerprint. This last security notion thus suggest the use of a blind protocol and a threshold scheme.

Remark 3. Contrary to blind signatures, user anonymity is not required; in our use-case, contact information must be joined with the final published fingerprint.

3 Assumptions

Our construction adapts the randomizable signature proposed by Pointcheval and Sanders [PS16], which relies on q-MSDH-1 [PS18]. Our scheme additionally requires indistinguishability, which implies another assumption; for this, we introduce q-DMSDH-1, which is decisional variant of q-MSDH-1, and prove it to hold in the generic bilinear group model.

Definition 4 (q-MSDH-1). *Let* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *be a bilinear group setting of type 3, with* g *(respectively* $\tilde{g}$*) a generator of* $\mathbb{G}_1$ *(respectively* $\mathbb{G}_2$*). Given* $(g^{x^i}, \tilde{g}^{x^i})_{0 \leq i \leq q}$ *along with* $(g^a, \tilde{g}^a, \tilde{g}^{a \cdot x})$ *for* $a, x \xleftarrow{\$} \mathbb{Z}_p^*$*, no adversary can output a tuple* $(w, P, h^{\frac{1}{x+w}}, h^{\frac{a}{P(x)}})$ *for some* $h \in \mathbb{G}_1^*$ *where* P *is a polynomial of degree at most* q *and* w *is a scalar such that* $(X + w)$ *and* $P(X)$ *are relatively prime.*

Definition 5 (q-DMSDH-1). *Let* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *be a bilinear group setting of type 3, with* g *(respectively* $\tilde{g}$*) a generator of* $\mathbb{G}_1$ *(respectively* $\mathbb{G}_2$*). Given* $(g^{x^i}, \tilde{g}^{x^i})_{0 \leq i < q}$ *along with* $(g^a, g^{a \cdot x}, \tilde{g}^a)$ *for* $a, x \xleftarrow{\$} \mathbb{Z}_p^*$*, and for any* (w, P) *where*

P is a polynomial of degree at most q and w is a scalar such that $(X + w)$ and $P(X)$ are relatively prime, no adversary can distinguish $(h^{\frac{1}{x+w}}, h^{\frac{a}{P(x)}})$ for some $h \in \mathbb{G}_1^*$ from a random pair of elements of $\mathbb{G}_1$.

Theorem 1. *q-DMSDH-1 holds in the generic bilinear group model.*

Proof. The computational assumption q-MSDH-1 from [PS18] gives $\tilde{g}^{a \cdot x} \in \mathbb{G}_2$ and expects the forged pair in $\mathbb{G}_1$, whereas the decisional version q-DMSDH-1 gives $g^{a \cdot x} \in \mathbb{G}_1$ and the challenge pair in $\mathbb{G}_1$. So, the group the pair belongs to determines what security guarantee we obtain (either unforgeability from q-MSDH-1 or indistinguishability from q-DMSDH-1). Thus, the reasoning for q-DMSDH-1 is very similar to that for q-MSDH-1. The full proof can be found in Appendix A.

4 Fingerprinting from Pointcheval-Sanders Signatures

In the following, we focus on our initial scenario with secret fingerprinting and public testing of plaintext-equality, for low-entropy messages. Our construction is heavily influenced by the assumption that it is possible to efficiently enumerate all the valid messages.

4.1 The Pointcheval-Sanders Signature Scheme

Our construction derives from Pointcheval-Sanders signatures [PS16,PS18]. We reproduce here the definition for the single-message version. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a type-3 pairing with $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p, and $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$. Then, we define the following procedures.

- $\mathsf{KeyGen}(1^k)$: $(\tilde{g}, x, y) \xleftarrow{\$} \mathbb{G}_2 \times \mathbb{Z}_p^2$, $\mathsf{sk} = (x, y)$ and $\mathsf{pk} = (\tilde{g}, \tilde{X} = \tilde{g}^x, \tilde{Y} = \tilde{g}^y)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: draw $h \xleftarrow{\$} \mathbb{G}_1^*$ and return $\sigma = (h, h^{x+ym})$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: return 1 if $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \tilde{X}\tilde{Y}^m) = e(\sigma_2, \tilde{g})$, else 0.

This signature scheme has been shown unforgeable in the sense of EUF–CMA under the interactive PS assumption [PS16], and in the sense of EUF–wCMA (non-adaptative) under the q-MSDH-1 assumption (where q is the bound on signing requests asked before the setup) [PS18]. The same levels of security are achieved when elements $g \in \mathbb{G}_1$ and $Y = g^y$ are included in the public key pk, as long as $X = g^x$ is kept private.

4.2 Fingerprinting Scheme with Public Plaintext-Equality Testing

Let $\mathcal{H}$ be a random oracle. We now propose a fingerprinting scheme where the fingerprinting procedure requires a secret key lk or rk, while testing is a public process (there is no testing key tk, or alternatively it is public).

- $\mathsf{KeyGen}(1^k)$: randomly draw $(g, \tilde{g}, x, y) \xleftarrow{\$} \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_p^2$, set $(X, \tilde{X}, Y, \tilde{Y}) \leftarrow (g^x, \tilde{g}^x, g^y, \tilde{g}^y)$, return $\mathsf{lk} = X$, $\mathsf{rk} = \tilde{X}$, and $\mathsf{pk} = (g, Y, \tilde{g}, \tilde{Y})$.

- $\mathsf{LFingerprint}(\mathsf{lk}, m)$: draw $u \xleftarrow{\$} \mathbb{Z}_p^*$, return $f_L = (g^u, (XY^{\mathcal{H}(m)})^u)$.
- $\mathsf{RFingerprint}(\mathsf{rk}, m)$: draw $u \xleftarrow{\$} \mathbb{Z}_p^*$, return $f_R = (\tilde{g}^u, (\tilde{X}\tilde{Y}^{\mathcal{H}(m)})^u)$.
- $\mathsf{Test}(f_L, f_R)$: return 1 if $f_{L,1}, f_{R,1} \neq 1_{\mathbb{G}_1}$ and $e(f_{L,1}, f_{R,2}) = e(f_{L,2}, f_{R,1})$, else 0.

4.3 Security of the Basic Scheme

Theorem 2. *Our fingerprinting scheme is FP−UF under q-MSDH-1 in the random oracle model, where q corresponds to the number of queries to the random oracle or to the fingerprinting oracles.*

Proof. We define the extended Pointcheval-Sanders signature scheme (EPS) as a variant of the PS signature scheme where pk includes Y, i.e. $\mathsf{pk} = (Y, \tilde{g}, \tilde{X}, \tilde{Y})$. We argue that EPS is EUF–wCMA secure under q-MSDH-1 in Lemma 1, and reduce the FP−UF security of our fingerprinting scheme to the EUF–wCMA security of EPS in Lemma 2.

Lemma 1. *If q-MSDH-1 holds, then EPS is EUF–wCMA where q is the number of queries to the signing oracle.*

Proof. We refer to the proof of theorem 10 from [PS18, Sect. 5.1, p. 330] where the challenger is given $(g^{x^i})_i$, $(\tilde{g}^{x^i})_i$ and $(g^a, \tilde{g}^a, \tilde{g}^{a \cdot x})$ and feeds the challenger with $\tilde{Y}_1 \leftarrow \tilde{g}^a$ and $\tilde{Y}_i \leftarrow \tilde{Y}_1^{u_i}$. To prove the EUF–wCMA security of the signature scheme when pk includes $(Y_i)_i$, it suffices to have the challenger also offer $Y_1 \leftarrow g^a$ and $Y_i \leftarrow Y_1^{u_i}$.

Lemma 2. *If EPS is EUF–wCMA, then our fingerprinting scheme is FP−UF. The number of queries to the signing oracle in EPS maps to the number of queries to the random oracle or to the fingerprinting oracles in our scheme.*

Proof. Let $\mathcal{A}$ be an adversary that breaks the FP−UF security of our scheme. Then, we create an adversary $\mathcal{B}$ that breaks the EUF–wCMA security of EPS. By symmetry of the left and right games, we assume that $\mathsf{Adv}^{\mathsf{FP-IND}}_{\Pi,L}(\mathcal{A})$ is non-negligible without loss of generality.

We will use $\mathcal{H}$ to "redirect" the queries from $\mathcal{A}$ towards predetermined values: $\mathcal{B}$ first draws $(m_i)_i \xleftarrow{\$} \mathbb{Z}_p^q$, submits the list of messages $(m_i)_i$ to the signing challenger, and will answer to the i-th original query to $\mathcal{H}$ (for some message M_i) with m_i.

In return, our adversary $\mathcal{B}$ is given $\mathsf{pk} = (Y, \tilde{g}, \tilde{X}, \tilde{Y})$ as well as signatures $(\sigma_i)_i$ for $(m_i)_i$, i.e. values such that $e(\sigma_{i,1}, \tilde{X}\tilde{Y}^{m_i}) = e(\sigma_{i,2}, \tilde{g})$ We need to output $(m^\star, \sigma^\star)$ such that $e(\sigma_1^\star, \tilde{X}\tilde{Y}^{m^\star}) = e(\sigma_2^\star, \tilde{g})$ where $m^\star$ is distinct from any queried m_i.

For this, we simulate the FP−UF game for $\mathcal{A}$ with $\mathsf{pk}' \leftarrow (g, Y, \tilde{g}, \tilde{Y})$, $\mathsf{rk} \leftarrow \tilde{X}$ as well as access to an oracle $\mathcal{L}$ which answer to queries M_i with σ_i. Then, $\mathcal{A}$ should output $(M^\star, f_L^\star)$ where $M^\star$ is distinct from any queried M_i. We also

require that $\mathsf{Test}(\mathsf{tk}, f_L^\star, f_R)$ for some $f_R \leftarrow \mathsf{RFingerprint}(\mathsf{rk}, m^\star) = 1$, i.e. such that $f_{L,1} \neq 1_{\mathbb{G}_1}$ and:

$$e\left(f_{L,1}, \left(\tilde{X}\tilde{Y}^{\mathcal{H}(M^\star)}\right)^u\right) = e\left(f_{L,2}, \tilde{g}^u\right)$$

for some u. Thus, $\sigma^\star = f_L^\star$ is a valid PS signature for $m^\star = \mathcal{H}(M^\star)$ with $m^\star$ distinct from any queried m_i.

Theorem 3. *Our fingerprinting scheme is FP−IND under q-DMSDH-1 in the random oracle model, where q corresponds to the number of queries to the random oracle or to the fingerprinting oracles.*

Proof. Let $\mathcal{A}$ be an adversary against $\mathsf{FP-IND}$, then we provide an adversary $\mathcal{B}$ against q-DMSDH-1. We assume that $\mathsf{Adv}_{\Pi,L}^{\mathsf{FP-IND}}(\mathcal{A})$ is non-negligible. Since the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ are symmetric, the same reasoning applies when $\mathsf{Adv}_{\Pi,R}^{\mathsf{FP-IND}}(\mathcal{A})$.

First, $\mathcal{B}$ is given $(g^{x^i})_{0\leq i\leq q}$, $(g^a, g^{a\cdot x}, \tilde{g}^a)$. Then, it draws $(m_i)_i \xleftarrow{\$} \mathbb{Z}_p^q$, $m \xleftarrow{\$} \mathbb{Z}_p$, sets $P = \prod_i (X + m_i)$, and submits (m, P) to the challenger, which answers with a pair σ which is either random or of the form $(h^{\frac{1}{x+m}}, h^{\frac{a}{P(x)}})$ for some $h \in \mathbb{G}_1$.

Now, $\mathcal{B}$ should be able to distinguish between these two cases. For this, $\mathcal{B}$ will run $\mathcal{A}$ while simulating the game for $\mathsf{FP-IND}$ by setting $g' \leftarrow g^{\prod_i (x+m_i)}$ and $\tilde{g'} \leftarrow \tilde{g}^{\prod_i (x+m_i)}$, using $(g^{x^i})_i$ and $(\tilde{g}^{x^i})_i$, as well as $X \leftarrow g^{a\cdot x}$, $Y \leftarrow g^a$, and $\tilde{Y} \leftarrow \tilde{g}^a$ to define the public key $\mathsf{pk} = (g', Y, \tilde{g'}, \tilde{Y})$ and the left-fingerprinting key $\mathsf{lk} = X$. This implicitly sets $x' = \frac{a\cdot x}{\prod_i (x+m_i)}$ and $y' = \frac{a}{\prod_i (x+m_i)}$.

To generate fingerprints for the q queried fingerprints, $\mathcal{B}$ sets the random oracle $\mathcal{H}$ to map the j-th original query M_j to m_j, and the right-fingerprinting oracle $\mathcal{R}$ to return $((\tilde{g'}^{\prod_{i\neq j}(x+m_i)})^{u_j}, (\tilde{g'}^a)^{u_j})$. One may verify that this is a valid right-fingerprint for M_j.

Finally, $\mathcal{A}$ outputs (M'_0, M'_1), and $\mathcal{B}$ draws $b \leftarrow \{0, 1\}$. We would now like to set $\mathcal{H}(M'_b)$ to m, but $\mathcal{A}$ may have queried the random oracle on this value before. Thus, on any query M_j, $\mathcal{H}$ will additionally guess with probability $\frac{1}{q}$ that $M_j = M'_b$ and accordingly set $\mathcal{H}(M_j)$ to m instead of m_j. $\mathcal{B}$ can then check its guess when $\mathcal{A}$ outputs (M'_0, M'_1), and abort if it was incorrect; this implies a penalty of a factor q to the probability that $\mathcal{B}$ wins the q-DMSDH-1 game.

Now, since $\mathcal{H}(M'_b) = m$, if σ is of the form $(h^{\frac{1}{x+m}}, h^{\frac{a}{P(x)}})$, then it is a valid left-fingerprint for M_b. Otherwise, it provides no information about b to the adversary. Thus, $\mathcal{B}$ answers the final request of $\mathcal{A}$ with σ, and, if $\mathcal{A}$ guesses b correctly, then $\mathcal{B}$ guesses that σ is of the form $(h^{\frac{1}{x+m}}, h^{\frac{a}{P(x)}})$; otherwise, that it is a random pair.

4.4 Improving the Privacy of the User

Since the left and right fingerprintings work in similar ways, we will only present the protocols for left fingerprinting.

Against the Fingerprinter Without the Final Fingerprint. In the naive construction above, the user sends the message in the clear to get back the fingerprint. In order to extend user privacy to the fingerprinters, we propose a blinded version, as in [PS16]:

1. the user draws $r \xleftarrow{\$} \mathbb{Z}_p$ and sends $C \leftarrow Y^m g^r$;
2. the user runs a Zero-Knowledge Proof of Knowledge (ZKPoK) of m, r such that $C = Y^m g^r$;
3. the fingerprinter draws $u \xleftarrow{\$} \mathbb{Z}_p$ and sends back $\alpha \leftarrow (g^u, (XC)^u)$;
4. the user sets $f_1 \leftarrow \alpha_1$, $f_2 \leftarrow \alpha_2 \cdot \alpha_1^{-r}$.

This protocol is perfectly blind to the fingerprinter, since his view is just the perfectly hiding Pedersen commitment [Ped92] and a ZK protocol, which do not leak any information about m. Hence the privacy of the user. With an extractable ZKPoK, it is possible to prove the security of this blinded version, as in [PS16].

Against the Fingerprinter with the Final Fingerprint. With the protocol presented above, if the fingerprinter gains access to the final fingerprint f, their ability to create fingerprints for arbitrary messages and the publicness of the testing key let them retrieve the message. In order to block exhaustive searches, we amend the protocol by splitting the fingerprinter into n parties, using secret sharing of the fingerprinting key. For some threshold k, no collusion of less than k parties can generate fingerprints; equivalently, having access to up to $k-1$ shares of the fingerprinting key does not reveal more than being a common user.

The threshold version makes use of Shamir's secret sharing [Sha79] to split the secret scalar x into n shares x_i (for each sub-fingerprinter F_i), and we note $X_i = g^{x_i}$. This way, for any qualified subset of F_i (with at least k shares), there are public coefficients λ_i (Lagrange coefficients) such that $x = \sum \lambda_i x_i$, and then $\prod X_i^{\lambda_i} = X$. A group of k sub-fingerprinters interacts as follows with the user:

1. the user draws $r \xleftarrow{\$} \mathbb{Z}_p$ and broadcasts $C \leftarrow Y^m g^r$;
2. the user sends a NIZKPoK of m, r such that $C = Y^m g^r$;
3. each F_i draws $u_i \xleftarrow{\$} \mathbb{Z}_p$ and broadcasts $\alpha_{i,1} \leftarrow g^{u_i}$;
4. each F_i computes $G \leftarrow \prod \alpha_{j,1}^{\lambda_j}$, and sends back $\alpha_{i,2} \leftarrow G^{x_i} C^{u_i}$;
5. the user sets

$$f_1 \leftarrow G = g^u \qquad\qquad f_2 \leftarrow G^{-r} \prod \alpha_{i,2}^{\lambda_i} = (XY^m)^u$$

which implicitly defines $u = \sum \lambda_i u_i$.

First, one can easily see that this still preserves the privacy of the user, since, as before, C does not contain any information about m; neither does the NIZK. The final fingerprint f traces back to the user but the anonymity is not required: in our use case, it must be possible to contact the appropriate hospital when a match is found. The important property is the privacy of m: no subset of less than k sub-fingerprinters can guess the conduct an exhaustive search.

Of course, we have to prove this still preserves fingerprinter privacy, or more precisely this does not leak private information of honest sub-fingerprinters to

corrupted ones. To this aim, we show that the view of any (static) subset of corrupted sub-fingerprinters can be simulated from the same information $\alpha = (\alpha_1, \alpha_2)$ as the one output by the fingerprinter in the centralized protocol.

Let us assume that the corrupted sub-fingerprinters are $\mathsf{F}_1, \ldots, \mathsf{F}_c$, and the honest ones are $\mathsf{F}_{c+1}, \ldots, \mathsf{F}_k$ (where $c < k$), and the simulator has drawn $v_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = c+1, \ldots, k$: the corrupted players send $\alpha_{i,1}$ for $i = 1, \ldots, c$, and the simulator draws $u_i \xleftarrow{\$} \mathbb{Z}_p$ and generates $\alpha_{i,1} \leftarrow g^{u_i}$ for $i = c+1, \ldots, k-1$, while $\alpha_{k,1} \leftarrow (\alpha_1 / \prod_{i=1}^{k-1} \alpha_{i,1}^{\lambda_i})^{1/\lambda_k}$. The simulator also sets $G \leftarrow \alpha_1$, and computes $\alpha_{i,2} \leftarrow G^{v_i} C^{u_i}$ for $i = c+1, \ldots, k-1$, while $\alpha_{k,2} \leftarrow (\alpha_2 / \prod_{i=1}^{k-1} \alpha_{i,2}^{\lambda_i})^{1/\lambda_k}$.

Since no information is known about the actual secret values x_i, and the values v_i are indistinguishable from the real secret, all the simulated elements are perfectly indistinguishable from a real execution, under the condition that the corrupted sub-fingerprinters are honest-but-curious (and the subset of honest players remains the same: static corruptions). Indeed, in this protocol, no verifiability is required about the sub-fingerprinters: they are trusted to compute correctly, even if they try to learn more information.

4.5 Verifiability

If one wants to consider malicious sub-fingerprinters, verifiability is required for the user (private verifiability). An additional improvement can be reached: one could avoid fake or copies of fingerprints posted by malicious users in the database, by using a proof of knowledge of the fingerprinted message (public verifiability). To achieve this, sub-fingerprinters first need to publish a commitment $C_i = g^{x_i} Y^{t_i}$ of their secret shares x_i during key generation. This is a perfectly hiding commitment, and the binding property relies on secrecy of the discrete logarithm of Y in basis g.

Private Verifiability. For the former case, verifiability can be enforced during the original process of creating the fingerprint, with the additional verification of a NIZK Proof of Existence of u_i and a NIZK Proof of Knowledge of x_i and t_i such that $\alpha_{i,1} = g^{u_i}$ and $\alpha_{i,2} = G^{x_i} C^{u_i}$. The proofs can be efficiently done with Schnorr-like proofs.

Public Verifiability. In order to avoid fake fingerprints or copies, the user should prove their validity (to avoid fake ones) and freshness (to avoid copies). A non-malleable NIZK could solve this challenge: in addition to $f = (f_1, f_2)$, and the NIZKs provided by the sub-fingerprinters, the user sends a NIZK Proof of Knowledge of m and r such that $\alpha_2 / f_2 = \alpha_1^r$ and $C = Y^m g^r$. In order to guarantee non-malleability or replay attacks, the user includes his own identity in the challenge computation (signature of knowledge).

4.6 Full Protocol

Let us now fully describe the resulting protocol, with an optimized NIZK for the public verifiability: the fingerprinters F_i, for $i = 1, \ldots, n$, jointly generate a Shamir's secret sharing of a random scalar secret x. They each own a share x_i, and publish a commitment $C_i = g^{x_i} Y^{t_i}$, for a random scalar t_i. In order to get a fingerprint on a message m, the user (with identity Id) contacts a subset of k sub-fingerprinters:

1. the user draws $r \xleftarrow{\$} \mathbb{Z}_p$ and broadcasts $C \leftarrow Y^m g^r$;
2. the user sends an (extractable) NIZKPoK of m, r such that $C = Y^m g^r$;
3. each F_i draws $u_i \xleftarrow{\$} \mathbb{Z}_p$ and sends back $\alpha_{i,1} \leftarrow g^{u_i}$;
4. each F_i computes $G \leftarrow \prod \alpha_{j,1}^{\lambda_j}$, and sends back $\alpha_{i,2} \leftarrow G^{x_i} C^{u_i}$;
5. each F_i starts a NIZK for u_i, x_i and t_i such that

$$\alpha_{i,1} = g^{u_i} \qquad \alpha_{i,2} = G^{x_i} C^{u_i}.$$

   More precisely, it draws $u'_i, x'_i, t'_i \xleftarrow{\$} \mathbb{Z}_p$ and sends

$$A_{i,1} = g^{u'_i} \qquad A_{i,2} = G^{x'_i} C^{u'_i};$$

6. the user generates

$$\alpha_1 \leftarrow G = \prod \alpha_{i,1}^{\lambda_i} = g^u \qquad \alpha_2 \leftarrow \prod \alpha_{i,2}^{\lambda_i} = (XC)^u$$
$$A_1 \leftarrow \prod A_{i,1}^{\lambda_i} = g^{u'} \qquad A_2 \leftarrow \prod A_{i,2}^{\lambda_i} = (XC)^{u'},$$

   where $u = \sum \lambda_i u_i$ and $u' = \sum \lambda_i u'_i$, as well as

$$f_1 \leftarrow \alpha_1 \qquad f_2 \leftarrow G^{-r} \alpha_2$$

   and starts the NIZK for m and r such that $\alpha_2 / f_2 = \alpha_1^r$ and $C = Y^m g^r$, with random r' and m':

$$B_1 \leftarrow \alpha_1^{r'} \qquad B_2 \leftarrow Y^{m'} g^{r'}$$

   and publishes the challenge $e = \mathcal{H}(\mathsf{Id}, C, f_1, f_2, A_1, A_2, B_1, B_2)$;
7. each F_i completes the NIZK with

$$u''_i \leftarrow u'_i - e u_i \qquad x''_i \leftarrow x'_i - e x_i$$

8. the user sets

$$u'' \leftarrow \sum \lambda_i u''_i \qquad x'' \leftarrow \sum \lambda_i x''_i$$

   which satisfy

$$g^{u''} = A_1 \alpha_1^{-e} \qquad G^{x''} C^{u''} = A_2 \alpha_2^{-e}$$

   and completes his NIZK with

$$m'' \leftarrow m' - em \qquad r'' \leftarrow r' - er$$

   which satisfy

$$\alpha_1^{r''} = B_1 (\alpha_2 / f_2)^{-e} \qquad Y^{m''} g^{r''} = B_2 C^{-e}.$$

The final fingerprint $f = (f_1, f_2)$ is published along with the intermediate values (α_2, C), the challenge e and the exponents (u'', x'', m'', r''), which constitute a proof that can be verified by checking that $e = \mathcal{H}(\mathsf{Id}, C, f_1, f_2, A_1, A_2, B_1, B_2)$, where the missing elements can be recomputed as

$$A_1 \leftarrow g^{u''} f_1^e \qquad A_2 \leftarrow f_1^{x''} C^{u''} \alpha_2^e$$
$$B_1 \leftarrow f_1^{r''} (\alpha_2 / f_2)^e \qquad B_2 \leftarrow Y^{m''} g^{r''} C^e$$

This is just an optimization of the Fiat-Shamir heuristic of Schnorr-like proofs.

5 Conclusion

With this construction, we are able to propose a new kind of scheme that let us derive testable, privacy-preserving, fingerprints from arbitrary messages. This allows us to propose a solution to the initial motivation of organ donation, where the requirement of encrypting low-entropy messages in such a way that they could be publicly tested against each other seemed to imply highly-interactive protocols. In contrast, our construction allows plaintext-equality tests between fingerprints to be performed fully offline, while only their generation requires an interactive process, to guarantee some level of confidentiality despite the low min-entropy.

We hope that this solution will prove useful in practical applications and allow cryptography to be used in more numerous situations. It might also be feasible to design systems which requires fewer interactions, or rely on more mainstream assumptions.

Acknowledgments. This work was supported in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

A Proof of Theorem 1

Proof. We prove q-DMSDH-1 in the generic bilinear group model. The generic group model (not bilinear) was used by Victor Shoup in [Sho97] to assess more tightly the difficulty of computing the discrete logarithm and related problems. A vastly clarified introduction to this technique can be found in [Jag12]. The generic bilinear group model is presented in appendix A of [BBG05]. It is essentially a formal way to enumerate the values that an adversary can compute from a restricted number of inputs, using only the group laws.

We use the classical approach of simulating group operations by an oracle $\mathcal{G}$, which operates on arbitrary representations $(\xi_{i,1})_i$, $(\xi_{i,2})_i$, $(\xi_{T,i})_i$ of the elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ (respectively). The oracle is built such that all interactions are done without relation to the secret values, hence reducing the attack to a guess.

For instance, $\mathcal{G}(\times, \xi_{i,1}, \xi_{1,j})$ returns a representation of the product of the underlying values in $\mathbb{G}_1$. The oracle $\mathcal{G}$ similarly allows the adversary $\mathcal{A}$ to compute products in $\mathbb{G}_2$ and $\mathbb{G}_T$, evaluate the pairing e, and test two representations for the equality of the underlying values.

To simulate the operations, the oracle $\mathcal{G}$ stores the values known to the adversary $\mathcal{A}$ (at beginning, and following a request) into lists L_1, L_2 and L_T (for each group). To track how the adversary $\mathcal{A}$ obtained these values, we save with each representation $\xi_{\square,i}$ a polynomial $p_{\square,i}$ corresponding to the operations used to compute the value. The representations used are not important, and the reader must simply remember that a new random representation is generated for each new computed value; testing whether the value is fresh or not is done by searching the polynomial in the relevant list L_1, L_2 or L_T.

The values initially provided to the adversary $\mathcal{A}$ are:

- in $\mathbb{G}_1$: $(g^{x^i})_{0 \le i \le q}$, g^a, $g^{a \cdot x}$, $h^{\frac{1}{x+w}}$, $h^{\frac{a}{P(x)}}$
- in $\mathbb{G}_2$: $(\tilde{g}^{x^i})_{0 \le i \le q}$, $\tilde{g}^a$

To simulate operations over these elements, we set r such that $h = g^r$ and introduce the indeterminate values $\bar{x}$, $\bar{a}$, $\bar{r}$. Then, we initialize $L_1 = \{\bar{x}^i\}_i \cup \{\bar{a}, \bar{a}\bar{x}, \frac{\bar{r}}{\bar{x}+w}, \frac{\bar{a}\cdot\bar{r}}{P(\bar{x})}\}$, $L_2 = \{\bar{x}^i\}_i \cup \{\bar{a}\}$ and $L_T = \varnothing$ (along with arbitrary representations), and set:

- $\mathcal{G}(\times, \xi_{\square,i}, \xi_{\square,j})$: append $p_{\square,i} + p_{\square,j}$ to $L_\square$
- $\mathcal{G}(=, \xi_{\square,i}, \xi_{\square,j})$: return whether $p_{\square,i} = p_{\square,j}$
- $\mathcal{G}(e, \xi_{1,i}, \xi_{2,j})$: append $p_{1,i} \times p_{2,j}$ to L_T

Remark 4. Comparing the representations directly is equivalent to calling the group oracle for testing, because the representations are generated so as to be equal when the corresponding polynomials are equal

We now have to show two things: the simulation does not allow the adversary to distinguish between $(h^{\frac{1}{x+w}}, h^{\frac{a}{P(x)}})$ and a pair of random elements from $\mathbb{G}_1$; the simulation is indistinguishable from the initial game.

Indistinguishability in Simulation. Since representations are opaque, the adversary can only obtain information from testing two values for equality (either of representations or through the group oracle $\mathcal{G}$).

Comparing elements of $\mathbb{G}_1$. Consider a comparison of $\xi_{1,i}$ to $\xi_{1,j}$; the difference of their polynomials, $p_{1,i} - p_{1,j}$, is of the form:

$$\sum_i \left(C_x^{(i)} \bar{x}^i + C_a \bar{a} + C_{ax} \bar{a}\bar{x} + C_1 \frac{\bar{r}}{\bar{x}+w} + C_2 \frac{\bar{a}\cdot\bar{r}}{P(\bar{x})} \right)$$

as a polynomial in $\bar{r}$, the linear term implies that, if this polynomial were equal to zero, then:

$$C_1 P(\bar{x}) + C_2 \bar{a}(\bar{x}+w) = 0$$

as a polynomial in $\bar{a}$, this implies $C_1 = C_2 = 0$. Thus, the polynomial does not depend on the challenge pair.

Comparing elements of $\mathbb{G}_2$. Elements in $\mathbb{G}_2$ do not depend on the challenge pair.

Comparing elements of $\mathbb{G}_T$. Since L_T starts out empty, a comparison of $\xi_{T,i}$ to $\xi T, j$ will correspond to polynomials whose difference $p_{T,i} - p_{T,j}$ is the sum of products of one element from $\mathbb{G}_1$ and one element from G_2, thus of the form:

$$\sum_i \left(Q(\bar{x}) + C_{i,a}\bar{a} + C_{i,ax}\bar{a}\bar{x} + C_{i,1}\frac{\bar{r}}{\bar{x}+w} + C_{i,2}\frac{\bar{a}\cdot\bar{r}}{P(\bar{x})} \right) \times \left(R(\bar{x}) + \tilde{C}_{i,a}\bar{a} \right)$$

where Q and R are polynomials of degrees at most q. As a polynomial in $\bar{r}$, if this were the zero polynomial, then the linear term would imply that:

$$\sum_i \left(C_{i,1}P(\bar{x}) + C_{i,2}\bar{a}(\bar{x}+w) \right) \times \left(R(\bar{x}) + \tilde{C}_{i,a}\bar{a} \right) = 0$$

as a polynomial in $\bar{a}$, then the linear term would imply that:

$$\sum_i \left(C_{i,1}P(\bar{x})\tilde{C}_{i,a} + C_{i,2}(\bar{x}+w)R(\bar{x}) \right) = 0$$

that is, $CP(\bar{x}) + S(\bar{x})(\bar{x}+w) = 0$ for C a constant and S a polynomial. Since $P(\bar{x})$ and $(\bar{x}+w)$ are relatively prime, this means that $C = 0$ and $S = 0$ and thus that the original equation does not depend on the challenge pair.

Undistinguishability of Simulation. Let $q_{\mathcal{G}}$ be the number of queries to the group oracle $\mathcal{G}$. The simulation is undistinguishable from the original game unless the adversary assembles two distinct polynomials (p, q) with $(p-q)(x, a, r) = 0$.

The adversary can adaptively test whether (x, a, r) is a root of one of the at most $q' = (5 + 2q + q_{\mathcal{G}})^2/2$ differences of polynomials of degrees at most $d = 2q$. Per the Schwartz-Zippel lemma, which states that a multivariate polynomial of degree d has at most d roots, this is equivalent to testing whether (x, a, r) pertains to one of q' subsets of $\mathbb{Z}_p^3$ of sizes at most d. Finally, the probability of adaptively finding such subsets is bounded above by $\frac{q' \cdot d}{p^3}$, which is negligible.

References

[BBG05] Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015 (2005). http://eprint.iacr.org/2005/015

[BBO06] Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. Cryptology ePrint Archive, Report 2006/186 (2006). http://eprint.iacr.org/2006/186

[BDOP04] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30

[CFGL12] Canard, S., Fuchsbauer, G., Gouget, A., Laguillaumie, F.: Plaintext-checkable encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 332–348. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_21

[Cha82] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 199–203. Plenum Press, New York (1982)

[FNP04] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_1

[GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)

[Jag12] Jager, T.: Black-Box Models of Computation. Vieweg+Teubner Verlag, Wiesbaden (2012)

[LZL13] Lu, Y., Zhang, R., Lin, D.: Stronger security model for public-key encryption with equality test. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 65–82. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36334-4_5

[Ped92] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9

[PS16] Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_7

[PS18] Pointcheval, D., Sanders, O.: Reassessing security of randomizable signatures. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 319–338. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_17

[Sha79] Shamir, A.: How to share a secret. Commun. Assoc. Comput. Mach. **22**(11), 612–613 (1979)

[Sho97] Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_18

[SWP00] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society Press, May 2000

[YTHW10] Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic public key encryption with equality test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_9