BETA: Biometric-Enabled Threshold Authentication

Shashank Agrawal[1], Saikrishna Badrinarayanan[2], Payman Mohassel[3], Pratyay Mukherjee[2], and Sikhar Patranabis[2]

[1] Western Digital, shashank.agrawal@wdc.com
[2] Visa Research, {sabadrin, pratmukh, sipatran}@visa.com
[3] Facebook, payman.mohassel@gmail.com

Abstract. In the past decades, user authentication has been dominated by server-side password-based solutions that rely on "what users know". This approach is susceptible to breaches and phishing attacks, and poses usability challenges. As a result, the industry is gradually moving to biometric-based client-side solutions that do not store any secret information on servers. This shift necessitates the safe storage of biometric templates and private keys, which are used to generate tokens, on user devices.

We propose a new generic framework called *Biometric Enabled Threshold Authentication* (BETA) to protect sensitive client-side information like biometric templates and cryptographic keys. Towards this, we formally introduce the notion of *Fuzzy Threshold Tokenizer* (FTT) where an initiator can use a "close" biometric measurement to generate an authentication token if at least t (the threshold) devices participate. We require that the devices only talk to the initiator, and not to each other, to capture the way user devices are connected in the real world. We use the universal composability (UC) framework to model the security properties of FTT, including the unforgeability of tokens and the privacy of the biometric values (template and measurement), under a *malicious* adversary. We construct *three* protocols that meet our definition.

Our first two protocols are general feasibility results that work for *any* distance function, *any* threshold t and tolerate the *maximal* (i.e. $t-1$) amount of corruption. They are based on *any* two round UC-secure multi-party computation protocol in the standard model (with a CRS) and threshold fully homomorphic encryption, respectively. We show how to effectively use these primitives to build protocols in a constrained communication model with just four rounds of communication.

For the third protocol, we consider inner-product based distance metrics (cosine similarity, Euclidean distance, etc.) specifically, motivated by the recent interest in its use for face recognition. We use Paillier encryption, efficient NIZKs for specific languages, and a simple garbled circuit to build an efficient protocol for the common case of $n = 3$ devices with one compromised.

1 Introduction

Traditionally, password-based authentication has been the dominant approach for authenticating users on the Internet, by relying on "what users know". However, this approach has its fair share of security and usability issues. It typically requires the servers to store a (salted) hash of all passwords, making them susceptible to offline dictionary attacks. Indeed, large-scale password breaches in the wild are extremely common [6,8]. Passwords also pose challenging usability problems. High entropy passwords are hard to remember by humans, while low entropy passwords provide little security, and research has shown that introducing complex restrictions on password choices can backfire [39, Sec A.3].

There are major ongoing efforts in the industry to address some of these issues. For example, "unique" biometric features such as finger-print [4], facial scans [1], and iris scans [9] are increasingly popular first or second factor authentication mechanisms for logging into devices and applications. Studies show that biometrics are much more user-friendly [2], particularly on mobile devices, as users do not have to remember or enter any secret information. At the same time, a (server-side) breach of biometric data is much more damaging because, unlike passwords, there is no easy way to change biometric information regularly.

Therefore, the industry is shifting away from transmitting or storing user secrets on the server-side. For example, biometric templates and measurements are stored and processed on the client devices where the matching also takes place. A successful match then unlocks a private signing key for a digital signature scheme which is used to generate a token on a fresh challenge. Instead of the user data, the token is transmitted to the server, who only stores a public verification key to verify the tokens. (Throughout the paper, we shall use the terms token and signature interchangeably.) Thus, a server breach does not lead to a loss of sensitive user data.

Most prominently, this is the approach taken by the FIDO Alliance [3], the world's largest industry-wide effort to enable an interoperable ecosystem of hardware-, mobile- and biometric-based authenticators that can be used by enterprises and service providers. This framework is also widely adopted by major Internet players and incorporated into all major browsers in the form of W3C standard Web Authentication API [10].

Hardware-based protection. With biometric data and private keys (for generating tokens) stored on client devices, a primary challenge is to securely protect them. As pointed out before, this is particularly crucial with biometrics since unlike passwords they are not replaceable. The most secure approach for doing so relies on hardware-based solutions such as secure enclaves [5] that provide physical separation between secrets and applications. However, secure hardware is not available on all devices, can be costly to support at scale, and provides very little programmability.

Software-based protection. Software-based solutions such as white-box cryptography are often based on ad-hoc techniques that are regularly broken [11]. The

provably secure alternative, i.e. cryptographic obfuscation [13,37], is not yet practical for real-world use-cases.

A simple alternative approach is to apply "salt-and-hash" techniques, often used to protect passwords, to biometric templates before storing them on the client device. Here, naïve solutions fail because biometric matching is almost always a fuzzy match that checks whether the distance between two vectors is above a threshold or not.

Using fuzzy extractors. It is tempting to think that a better way to implement the hash-and-salt approach for biometric data is through a cryptographic primitive known as *fuzzy extractor* [33,21]. However, as also discussed by Dupont et al. [34], this approach only works for high-entropy biometric data and is susceptible to offline dictionary attacks.

Distributed cryptography to the rescue. Our work is motivated by the fact that most users own and carry *multiple devices* (laptop, smart-phone, smart-watch, etc.) and have other IoT devices around when authenticating (smart TV, smart-home appliances, etc.). We introduce a new framework for client-side biometric-based authentication that securely distributes both the biometric template as well as the secret signing key among multiple devices. These devices can collectively perform biometric matching and token generation without ever reconstructing the template or the signing key on any one device. We refer to this framework as *Biometric Enabled Threshold Authentication* (BETA for short) and study it at length in this paper.

Before diving deeper into the details, we note that while our primary motivation stems from a client-side authentication mechanism, our framework is quite *generic* and can be used in other settings. For example, it can also be used to protect biometric information on the *server-side* by distributing it among multiple servers who perform the matching and token generation (e.g., for a *single sign-on* authentication token) in a fully distributed manner.

1.1 Our Contributions

To concretely instantiate our framework BETA, we formally introduce the notion of *fuzzy threshold tokenizer* (FTT). We provide a universally composable (UC) security definition for FTT and design several protocols that realize it. We first briefly describe the notion of a Fuzzy Threshold Tokenizer.

Fuzzy Threshold Tokenizer. Consider a set of n parties/devices, a distribution $\mathcal{W}$ over vectors in $\mathbb{Z}_q^\ell$, a threshold t on the number of parties, a distance predicate Dist and an unforgeable threshold signature scheme TS. Initially, in a *global setup* phase, a user generates some public and secret parameters (in a trusted setting), and distributes them amongst the n devices she owns. Further, she also runs the setup of the scheme TS and secret shares the signing key amongst the devices. In an *enrollment* phase, user samples a biometric template $\vec{\mathbf{w}} \in \mathbb{Z}_q^\ell$ according to $\mathcal{W}$ and securely shares it amongst all the devices. Any set

of t devices can, together, completely reconstruct the biometric template $\overrightarrow{\mathbf{w}}$ and the signing key of the threshold signature scheme. Then, during an online *sign on session*, an initiating device P, with a candidate biometric measurement $\overrightarrow{\mathbf{u}}$ as input, can interact in a protocol with a set $\mathcal{S}$ of $(t-1)$ other devices. At the end of this, if $\overrightarrow{\mathbf{u}}$ is "close enough" to the template $\overrightarrow{\mathbf{w}}$ (with respect to distance predicate Dist), the initiating device P obtains a token (signature) on a message of its choice.

It is important to note that we do not allow the other participating $(t-1)$ devices to interact amongst themselves[4] and all communication goes through the initiating device P. This is a critical requirement on the communication model for FTT since in a typical usage scenario, one or two primary devices (e.g., a laptop or a smart-phone) play the role of the initiating device and all other devices are only paired/connected to the primary device. (These devices may not even be aware of the presence of other devices.) Indeed, this requirement makes the design of constant-round FTT protocols significantly more challenging. Further, in any round of communication, we only allow unidirectional exchange of messages, i.e., either P sends a message to some subset of the other $(t-1)$ devices or vice versa.

Security definition. Consider a probabilistic polynomial time adversary $\mathcal{A}$ that corrupts a set T of devices where $|T| < t$. Informally, the security properties that we wish to capture in an FTT scheme are as follows:

(i) *Privacy of biometric template:* From any sign on session initiated by a corrupt device, $\mathcal{A}$ should not be able to learn any information about the biometric template $\overrightarrow{\mathbf{w}}$ apart from just the output of the predicate $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}})$ for its choice of measurement $\overrightarrow{\mathbf{u}}$. If the sign on session was initiated by an honest device, $\mathcal{A}$ should learn no information about $\overrightarrow{\mathbf{w}}$. Crucially, we do not impose any restriction on the entropy of the distribution from which the template is picked.

(ii) *Privacy of biometric measurement:* For any sign on session initiated by an honest device, $\mathcal{A}$ should learn no information whatsoever about the measurement $\overrightarrow{\mathbf{u}}$.

(iii) *Token unforgeability*: $\mathcal{A}$ should not be able to compute a valid token (that verifies according to the threshold signature scheme TS) unless it initiated a sign on session on behalf of a corrupt party with a measurement $\overrightarrow{\mathbf{u}}$ such that $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$. Furthermore, $\mathcal{A}$ should only be able to compute *exactly one* token from each such session.

Our *first contribution* is a formal modeling of the security requirements of a fuzzy threshold tokenizer via a real-ideal world security definition in the universal composability (UC) framework [26]. We refer the reader to Section 4 for the formal definition and a detailed discussion on its intricacies.

Our next contribution is a design of several protocols that realize this primitive.

[4]Note that corrupt parties can of course freely interact amongst themselves.

Protocol-1(π^{mpc}). Given any threshold signature scheme TS, for any distance measure Dist, any n, t, we construct a four round[5] UC-secure FTT protocol π^{mpc}. Our construction is based on any two-round (over a broadcast channel) UC-secure multi-party computation (MPC) protocol [45,49,38,15] in the CRS model that is secure against up to all but one corruption along with other basic primitives. π^{mpc} tolerates up to $(t-1)$ (which is maximal) malicious devices.

Protocol-2 (π^{tfhe}). Given any threshold signature scheme TS, for any distance measure Dist, any n, t, we construct a four round UC-secure FTT protocol π^{tfhe}. Our construction is based on any t out of n threshold fully homomorphic encryption scheme (TFHE) and other basic primitives. Like π^{mpc}, this protocol is secure against $(t-1)$ malicious devices.

The above two feasibility results are based on two incomparable primitives (two round MPC and threshold FHE). On the one hand, two-round MPC seems like a stronger notion than threshold FHE. But, on the other hand, two-round MPC is known from a variety of assumptions like LWE/DDH/Quadratic Residuosity, while threshold FHE is known only from LWE. Further, the two protocols have very different techniques which may be of independent interest.

Protocol-3 (π^{ip}). We design the third protocol π^{ip} specifically for the cosine similarity distance metric, which has recently been shown to be quite effective for face recognition (CosFace [56], SphereFace [43], FaceNet [54]). We pick a threshold of three for this protocol as people nowadays have at least three devices on them most of the time (typically, a laptop, a smart-phone and a smart-watch). π^{ip} is secure in the random oracle model as long as at most one of the devices is compromised. We use Paillier encryption, efficient NIZKs for specific languages, and a simple garbled circuit to build an efficient four-round protocol.

Efficiency analysis of π^{ip}. Finally, we perform a concrete efficiency analysis of our third protocol π^{ip}. We assume that biometric templates and measurements have ℓ features (or elements) and every feature can be represented with m bits. Let λ denote the computational security parameter and s denote the statistical security parameter. In the protocol π^{ip}, we use Paillier encryption scheme to encrypt each feature of the measurement and its product with the shares of the template. The initiator device proves that the ciphertexts are well-formed and the features are of the right length. For Paillier encryption, such proofs can be done efficiently using only $O(\ell m)$ group operations [31,30].

[5] Recall that by one communication round, we mean a unidirectional/non-simultaneous message exchange channel over a *peer-to-peer* network. That is, in each round either the initiator sends messages to some subset of the other participating devices or vice versa. In contrast, one round of communication over a *broadcast* channel means that messages are being sent *simultaneously* by multiple (potentially all) parties connected to the channel and all of them receive all the messages sent in that round. All our FTT protocols use peer-to-peer channels which is the default communication model in this paper.

The other devices use the homomorphic properties of Paillier encryption to compute ciphertexts for inner-product shares and some additional values. They are sent back to the initiator but with a MAC on them. Then the other devices generate a garbled circuit that takes the MAC information from them and the decrypted ciphertexts from the initiator to compute if the cosine value exceeds a certain threshold. The garbled circuit constructed here only does 5 multiplications on numbers of length $O(m+\log \ell+s)$. Oblivious transfers can be preprocessed in the setup phase between every pair of parties so that the online phase is quite efficient (only symmetric-key operations). Furthermore, since only one of the two helping devices can be corrupt, only one device needs to transfer the garbled circuit [44], further reducing the communication overhead. (We have skipped several important details of the protocol here, but they do not affect the complexity analysis. See Section 2.3 for a complete overview of the protocol.)

An alternate design appropach is to use the garbled circuit itself to compute the inner-product. However, there are two disadvantages of this approach. First, it does not scale efficiently with feature vector length. The number of multiplications to be done inside the garbled circuit would be linear in the number of features, or the size of the circuit would be roughly $O(m^2\ell)$. This is an important concern because the number of features in a template can be very large (e.g., see Figure 1 in the NISTIR draft on Ongoing Face Recognition Vendor Test (FRVT) [7]). Second, the devices would have to prove in zero knowledge that the bits fed as input to the circuit match the secret shares of the template given to them in the enrollment phase. This incurs additional computational overheads.

1.2 Related Work

Fuzzy identity based encryption, introduced by Sahai and Waters [53], allows decrypting a ciphertext encrypted with respect to some identity id if the decryptor possesses the secret key for an identity that almost matches id. However, unlike FTT, the decryptor is required to know both identities and which positions match. Recall that one of our main goals is to distribute the biometric template across all devices so that no one device ever learns it.

Function secret sharing, introduced by Boyle et al. [22], enables to share the computation of a function f amongst several users. Another interesting related primitive is homomorphic secret sharing [23]. However, both these notions don't quite fit in our context because of the limitations on our communication model and the specific security requirements against a malicious adversary.

Secure multiparty computation protocols in the private simultaneous messages model [36,41,14] consider a scenario where there is a client and a set of servers that wish to securely compute a function f on their joint inputs wherein the communication model only involves interaction between the client and each individual server. However, in that model, the adversary can either corrupt the client or a subset of servers but not both.

The work of Dupont et al. [34] construct a fuzzy password authenticated key exchange protocol where each of the two parties have a password with low entropy. At the end of the protocol, both parties learn the shared secret key

only if the two passwords are "close enough" with respect to some distance measure. In our work, we consider the problem of generating signatures and also multiple parties. Another crucial difference is that in their work, both parties hold a copy of the password whereas in our case, the biometric template is distributed between parties and therefore is never exposed to any party. There is also a lot of work on distributed password authenticated key exchange [16] (and the references within) but their setting considers passwords (and so, equality matching) and not biometrics.

There has been a lot of work in developing privacy-preserving ways to compare biometric data [25,17,32] but it has mostly focused on computing specific distance measures (like Hamming distance) in the two-party setting where each party holds a vector. There has also been some privacy-preserving work in the same communication model as ours [29,42,19] but it has mainly focused on private aggregation of sensitive user data.

Open Problems. We leave it as an open problem to define weaker game-based security definitions for FTT and to design more efficient protocols that satisfy those. We also leave it open to design FTT protocols that tolerate adaptive corruptions and/or support dynamic addition/deletion of parties and rotation of signature keys.

2 Technical Overview

2.1 MPC based protocol

Emulating General Purpose MPC. Our starting point is the observation that suppose all the parties could freely communicate, then any UC-secure MPC protocol against a malicious adversary in the presence of a broadcast channel would intuitively be very useful in the design of an FTT scheme if we consider the following functionality: the initiator P^* has input $(\mathsf{msg}, S, \overrightarrow{\mathbf{u}})$, every party $P_i \in S$ has input (msg, S), their respective shares of the template $\overrightarrow{\mathbf{w}}$ and the signing key. The functionality outputs a signature on msg to party P^* if $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$ and $|S| = t$. Recently, several works [45,50,24,38,15] have shown how to construct two round UC-secure MPC protocols in the CRS model in the presence of a broadcast channel from standard cryptographic assumptions. However, the issue with following this intuitive approach is that the communication model of our FTT primitive does not allow all parties to interact amongst each other - in particular, the parties in the set S can't directly talk to each other and all communication has to be routed through the initiator. Armed with this insight, our goal now is to emulate a two round MPC protocol π in our setting.

For simplicity, let us first consider $n = t = 3$. That is, there are three parties: P_1, P_2, P_3. Consider the case when P_1 is the initiator. Now, in the first round of our FTT scheme, P_1 sends msg to both parties. Then, in round 2, we have P_2 and P_3 send their round one messages of the MPC protocol π. In round 3 of our FTT scheme, P_1 sends its own round one message of the MPC protocol to

both parties. Along with this, P_1 also sends P_2's round one message to P_3 and vice versa. So now, at the end of round 3 of our FTT scheme, all parties have exchanged their first round messages of protocol π.

Our next observation is that since we care only about P_1 getting output, in the underlying protocol π, only party P_1 needs to receive everyone else's messages in round 2. Therefore, in round 4 of our FTT scheme, P_2 and P_3 can compute their round two messages based on the transcript so far and just send them to P_1. This will enable P_1 to compute the output of protocol π.

Challenges. Unfortunately, the above scheme is insecure. Note that in order to rely on the security of protocol π, we crucially need that for any honest party P_i, every other honest party receives the same first round message on its behalf. Also, we require that all honest parties receive the same messages on behalf of the adversary. In our case, since the communication is being controlled and directed by P_1 instead of a broadcast channel, this need not be true if P_1 was corrupt and P_2, P_3 were honest. Specifically, one of the following two things could occur: (i) P_1 can forward an incorrect version of P_3's round one message of protocol π to P_2 and vice versa. (ii) P_1 could send different copies of its own round 1 message of protocol π to both P_2 and P_3.

Signatures to Solve Challenge 2. To solve the first problem,we simply enforce that P_3 sends a signed copy of its round 1 message of protocol π which is forwarded by P_1 to P_2. Then, P_2 accepts the message to be valid if the signature verifies. In the setup phase, we can distribute a signing key to P_3 and a verification key to everyone, including P_2. Similarly, we can ensure that P_2's actual round 1 message of protocol π was forwarded by P_1 to P_3.

Pseudorandom Functions to Solve Challenge 2. Tackling the second problem is a bit trickier. The idea is instead of enforcing that P_1 send the same round 1 message of protocol π to both parties, we will instead ensure that P_1 learns their round 2 messages of protocol π only if it did indeed send the same round 1 message of protocol π to both parties. We now describe how to implement this mechanism. Let us denote msg_2 to be P_1's round 1 message of protocol π sent to P_2 and msg_3 (possibly different from msg_2) to be P_1's round 1 message of protocol π sent to P_3. In the setup phase, we distribute two keys k_2, k_3 of a pseudorandom function (PRF) to both P_2, P_3. Now, in round 4 of our FTT scheme, P_3 does the following: instead of sending its round 2 message of protocol π as is, it encrypts this message using a secret key encryption scheme where the key is $\mathsf{PRF}(k_3, \mathsf{msg}_3)$. Then, in round 4, along with its actual message, P_2 also sends $\mathsf{PRF}(k_3, \mathsf{msg}_2)$ which would be the correct key used by P_3 to encrypt its round 2 message of protocol π only if $\mathsf{msg}_2 = \mathsf{msg}_3$. Similarly, we use the key k_2 to ensure that P_2's round 2 message of protocol π is revealed to P_1 only if $\mathsf{msg}_2 = \mathsf{msg}_3$.

The above approach naturally extends for arbitrary n, t. by sharing two PRF keys between every pair of parties. There, each party encrypts its round 2 message of protocol π with a secret key that is an XOR of all the PRF evaluations. There are

additional subtle issues when we try to formally prove that the above protocol is UC-secure and we refer the reader to the full version [12] for more details about the proof.

2.2 Threshold FHE based protocol

The basic idea behind our second protocol is to use an FHE scheme to perform the distance predicate computation between the measurement $\overrightarrow{\mathbf{u}}$ and the template $\overrightarrow{\mathbf{w}}$. In particular, in the setup phase, we generate the public key pk of an FHE scheme and then in the enrollment phase, each party is given an encryption $\mathsf{ct}_{\overrightarrow{\mathbf{w}}}$ of the template. In the sign on phase, an initiator P^* can compute a ciphertext $\mathsf{ct}_{\overrightarrow{\mathbf{u}}}$ that encrypts the measurement and send it to all the parties in the set S which will allow them to each individually compute a ciphertext ct^* homomorphically that evaluates $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}})$. However, the first challenge is how to decrypt this ciphertext ct^*? In other words, who gets the secret key sk of the FHE scheme in the setup? If sk is given to all parties in S, then they can, of course, decrypt $\mathsf{ct}_{\overrightarrow{\mathbf{u}}}$ but that violates privacy of the measurement. On the other hand, if sk is given only to P^*, that allows P^* to decrypt $\mathsf{ct}_{\overrightarrow{\mathbf{w}}}$ violating privacy of the template.

Threshold FHE. Observe that this issue can be overcome if somehow the secret key is secret shared amongst all the parties in S in such a way that each of them, using their secret key share sk_i, can produce a partial decryption of ct^* that can then all be combined by P^* to decrypt ct^*. In fact, this is exactly the guarantee of threshold FHE. This brings us to the next issue that if only P^* learns whether $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$, how do the parties in S successfully transfer the threshold signature shares? (recall that the transfer should be conditioned upon Dist evaluating to 1) One natural option is, in the homomorphic evaluation of the ciphertext ct, apart from just checking whether $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$, perhaps the circuit could then also compute the partial signatures with respect to the threshold signature scheme if the check succeeds. However, the problem then is that, for threshold decryption, there must be a common ciphertext available to each party. In this case, however, each party would generate a partial signature using its own signing key share resulting in a different ciphertext and in turn preventing threshold decryption.

Partial Signatures. To overcome this obstacle, at the beginning of the sign-on phase, each party computes its partial signature σ_i and *information-theoretically* encrypts it via one-time pad with a uniformly sampled one-time key K_i. The parties then transfer the partial signatures in the same round in an encrypted manner without worrying about the result of the decryption. Now, to complete the construction, we develop a mechanism such that:

– Whenever the FHE decryption results in 1, P^* learns the set of one-time secret keys $\{K_i\}$ and hence reconstructs the set of partial signatures $\{\sigma_i\}$.

– Whenever the FHE decryption results in 0, P^* fails to learn any of the one-time secret keys, which in turn ensures that each of the partial signatures remains hidden from P^*.

To achieve that, we do the following: each party additionally broadcasts ct_{K_i}, which is an FHE encryption of its one-time secret key K_i, to every other party during the enrollment phase. Additionally, we use t copies of the FHE circuit being evaluated as follows: the i^{th} circuit outputs K_i if $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$ – that is, this circuit is homomorphically evaluated using the FHE ciphertexts $\mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \mathsf{ct}_{\overrightarrow{\mathbf{w}}}, \mathsf{ct}_{K_i}$. [6] Now, at the end of the decryption, if $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}})$ was indeed equal to 1, P^* learns the set of one-time keys $\{K_i\}$ via homomorphic evaluation and uses these to recover the corresponding partial signatures.

Consider the case where the adversary $\mathcal{A}$ initiates a session with a measurement $\overrightarrow{\mathbf{u}}$ such that $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 0$. Our security proof formally establishes that the adversary $\mathcal{A}$ learns no information about each one-time key K_i of the honest parties and hence about the corresponding signature share. At a high level, we exploit the simulation and semantic security guarantees of the threshold FHE scheme to: (a) simulate the FHE partial decryptions to correctly output 0 and (b) to switch each ct_{K_i} to be an encryption of 0. At this point, we can switch each K_i to be a uniformly random string and hence "unrecoverable" to $\mathcal{A}$. We refer the reader to Section 6 for more details.

NIZKs. One key issue is that parties may not behave honestly - that is, in the first round, P^* might not run the FHE encryption algorithm honestly and similarly, in the second round, each party might not run the FHE partial decryption algorithm honestly which could lead to devastasting attacks. To solve this, we require each party to prove honest behavior using a non-interactive zero knowledge argument (NIZK). Finally, as in the previous section, to ensure that P^* sends the same message $\mathsf{ct}_{\overrightarrow{\mathbf{u}}}$ to all parties, we use a signature-based verification strategy, which adds two rounds resulting in a four round protocol.

2.3 Cosine Similarity: single corruption

In this section, we build a protocol for a specific distance measure[7] (Cosine Similarity). It is more efficient compared to our feasibility results. On the flip side, it tolerates only one corruption: that is, our protocol is UC-secure in the Random Oracle model against a malicious adversary that can corrupt only one party. For two vectors $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}$, $\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = \frac{\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle}{||\overrightarrow{\mathbf{u}}|| \cdot ||\overrightarrow{\mathbf{w}}||}$ where $||\overrightarrow{x}||$ denotes the L^2-norm of the vector. $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$ if $\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) \geq d$ where d is chosen by Dist. Without loss of generality, assume that distribution $\mathcal{W}$ samples vectors $\overrightarrow{\mathbf{w}}$ with $||\overrightarrow{\mathbf{w}}|| = 1$. Then, we check if $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle > (d \cdot \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle)^2$ instead of

[6] Note that the creation and broadcasting of these ciphertexts can happen in parallel within a single round of communication between P^* and the other parties in the set S.

[7] Our construction can also be extended to work for the related Euclidean Distance function but we focus on Cosine Similarity in this section.

$\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) > d$. This syntactic change allows more flexibility.

Distributed Garbling. Our starting point is the following. Suppose we had $t = 2$. Then, we can just directly use Yao's [57] two party semi-honest secure computation protocol as a building block to construct a two round FTT scheme. In the enrollment phase, secret share $\overrightarrow{\mathbf{w}}$ into $\overrightarrow{\mathbf{w}}_1, \overrightarrow{\mathbf{w}}_2$ and give one part to each party. The initiator requests for labels via oblivious transfer (OT) corresponding to his share of $\overrightarrow{\mathbf{w}}$ and input $\overrightarrow{\mathbf{u}}$ while the garbled circuit, which has the other share of $\overrightarrow{\mathbf{w}}$ hardwired, reconstructs $\overrightarrow{\mathbf{w}}$, checks if $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle > (d \cdot \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle)^2$ and if so, outputs a signature. This protocol is secure against a malicious initiator who only has to evaluate the garbled circuit, if we use an OT protocol that is malicious secure in the CRS model. However, to achieve malicious security against the garbler, we would need expensive zero knowledge arguments that prove correctness of the garbled circuit. Now, in order to build an efficient protocol that achieves security against a malicious garbler and to work with threshold $t = 3$, the idea is to distribute the garbling process between two parties.

Consider an initiator P_1 interacting with parties P_2, P_3. We repeat the below process for any initiator and any pair of parties that it must interact with. For ease of exposition, we just consider P_1, P_2, P_3 in this section. Both P_2 and P_3 generate one garbled circuit each using shared randomness generated during setup and the evaluator just checks if the two circuits are identical. Further, both P_2 and P_3 get the share $\overrightarrow{\mathbf{w}}_2$ and a share of the signing key in the enrollment and setup phase respectively. Note that since the adversary can corrupt at most one party, this check would guarantee that the evaluator can learn whether the garbled circuit was honestly generated. In order to ensure that the evaluator does not evaluate both garbled circuits on different inputs, we will also require the garbled circuits to check that P_1's OT receiver queries made to both parties was the same. The above approach is inspired from the three party secure computation protocol of Mohassel et al. [44].

However, the issue here is that P_1 needs a mechanism to prove in zero knowledge that it is indeed using the share $\overrightarrow{\mathbf{w}}_1$ received in the setup phase as input to the garbled circuit. Moreover, even without this issue, the protocol is computationally quite expensive. For cosine similarity, the garbled circuit will have to perform a lot of expensive operations - for vectors of length ℓ, we would have to perform $O(\ell)$ multiplications inside the garbled circuit. As mentioned in the introduction, because the number of features in a template (ℓ) can be very large for applications like face recognition, our goal is to improve the efficiency and scalability of the above protocol by performing only a constant number of multiplications inside the garbled circuit.

Additive Homomorphic Encryption. Our strategy to build an efficient protocol is to use additional rounds of communication to offload the heavy computation outside the garbled circuit and also along the way, solve the issue of

the initiator using the right share $\overrightarrow{\mathbf{w}}_1$. In particular, if we can perform the inner product computation outside the garbled circuit in the first phase of the protocol, then the resulting garbled circuit in the second phase would have to perform only a constant number of operations. In order to do so, we leverage the tool of efficient additively homomorphic encryption schemes [48,35]. In our new protocol, in round 1, the initiator P_1 sends an encryption of $\overrightarrow{\mathbf{u}}$. P_1 can compute $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_1 \rangle$ by itself. Both P_2 and P_3 respond with encryptions of $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_2 \rangle$ computed homomorphically using the same shared randomness. P_1 can decrypt this to compute $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle$. The parties can then run the garbled circuit based protocol as above in rounds 3 and 4 of our FTT scheme: that is, P_1 requests for labels corresponding to $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle$ and $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle$ and the garbled circuit does the rest of the check as before. While this protocol is correct and efficient, there are still several issues.

Leaking Inner Product. The first problem is that the inner product $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle$ is currently leaked to the initiator P_1 thereby violating the privacy of the template $\overrightarrow{\mathbf{w}}$. To prevent this, we need to design a mechanism where no party learns the inner product entirely in the clear and yet the check happens inside the garbled circuit. A natural approach is for P_2 and P_3 to homomorphically compute an encryption of the result $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_2 \rangle$ using a very efficient secret key encryption scheme. In our case, just a one time pad suffices. Now, P_1 only learns an encryption of this value and hence the inner product is hidden, while the garbled circuit, with the secret key hardwired into it, can easily decrypt the one-time pad.

Input Consistency. The second major challenge is to ensure that the input on which P_1 wishes to evaluate the garbled circuit is indeed the output of the decryption. If not, P_1 could request to evaluate the garbled circuit on suitably high inputs of his choice, thereby violating unforgeability! In order to prevent this attack, P_2 and P_3 homomorphically compute not just $x = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_2 \rangle$ but also a message authentication code (mac) y on the value x using shared randomness generated in the setup phase. We use a simple one time mac that can be computed using linear operations and hence can be done using the additively homomorphic encryption scheme. Now, the garbled circuit also checks that the mac verifies correctly and from the security of the mac, P_1 can not change the input between the two stages. Also, we require P_1 to also send encryptions of $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle$ in round 1 so that P_2, P_3 can compute a mac on this as well, thereby preventing P_1 from cheating on this part of the computation too.

Ciphertext Well-formedness. Another important issue to tackle is to ensure that P_1 does indeed send well-formed encryptions. To do so, we rely on efficient zero knowledge arguments from literature [31,30] when instantiating the additively homomorphic encryption scheme with the Paillier encryption scheme [48]. For technical reasons, we also need the homomorphic encryption scheme to be circuit-private. We refer the reader to the full version [12] for more details. Observe that in our final protocol, the garbled circuit does only a constant number

of multiplications, which makes protocol computationally efficient and scalable.

Optimizations. To further improve the efficiency of our protocol, as done in Mohassel et al. [44], we will require only one of the two parties P_2, P_3 to actually send the garbled circuit. The other party can just send a hash of the garbled circuit and the initiator can check that the hash values are equal. We refer to Section 7 for more details on this and other optimizations.

3 Preliminaries

Let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ denote the n parties and λ the security parameter. Recall that the L^2 norm of a vector $\overrightarrow{\mathbf{x}} = (\overrightarrow{\mathbf{x}}_1, \ldots, \overrightarrow{\mathbf{x}}_n)$ is defined as $||\overrightarrow{\mathbf{x}}|| = \sqrt{\overrightarrow{\mathbf{x}}_1^2 + \ldots + \overrightarrow{\mathbf{x}}_n^2}$. $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle$ denotes the inner product between two vectors $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}$.

Definition 1. ***(Cosine Similarity)*** *For any two vectors $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \in \mathbb{Z}_q^\ell$, the Cosine Similarity between them is defined as follows:*

$$\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = \frac{\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}} \rangle}{||\overrightarrow{\mathbf{u}}|| \cdot ||\overrightarrow{\mathbf{w}}||}.$$

When using this distance measure, we say that $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$ if and only if $\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) \geq d$ where d is a parameter specified by $\mathsf{Dist}(\cdot)$.

3.1 Threshold Signature

Definition 2 (Threshold Signature [18]). Let $n, t \in \mathbb{N}$. A threshold signature scheme TS is a tuple of four algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Comb}, \mathsf{Ver})$ that satisfy the correctness condition below.

- $\mathsf{Gen}(1^\lambda, n, t) \to (\mathsf{pp}, \mathsf{vk}, [\![\mathbf{sk}]\!]_n)$. A randomized algorithm that takes n, t and the security parameter λ as input, and generates a verification-key vk and a shared signing-key $[\![\mathsf{sk}]\!]_n$.
- $\mathsf{Sign}(\mathsf{sk}_i, m) =: \sigma_i$. A deterministic algorithm that takes a mesage m and signing key-share sk_i as input and outputs a partial signature σ_i.
- $\mathsf{Comb}(\{\sigma_i\}_{i \in S}) =: \sigma/\perp$. A deterministic algorithm that takes a set of partial signatures $\{\mathsf{sk}_i\}_{i \in S}$ as input and outputs a signature σ or $\perp$ denoting failure.
- $\mathsf{Ver}(\mathsf{vk}, (m, \sigma)) =: 1/0$. A deterministic algorithm that takes a verification key vk and a candidate message-signature pair (m, σ) as input, and outputs 1 for a valid signature and 0 otherwise.

Correctness. For all $\lambda \in \mathbb{N}$, any $t, n \in \mathbb{N}$ such that $t \leq n$, all $(\mathsf{pp}, \mathsf{vk}, [\![\mathbf{sk}]\!]_n)$ generated by $\mathsf{Gen}(1^\lambda, n, t)$, any message m, and any set $S \subseteq [n]$ of size at least t, if $\sigma_i = \mathsf{Sign}(\mathsf{sk}_i, m)$ for $i \in S$, then $\mathsf{Ver}(\mathsf{vk}, (m, \mathsf{Comb}(\{\sigma_i\}_{i \in S}))) = 1$.

Definition 3 (Unforgeability). A threshold signatures scheme $\mathsf{TS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Comb}, \mathsf{Ver})$ is unforgeable if for all $n, t \in \mathbb{N}$, $t \leq n$, and any PPT adversary $\mathcal{A}$, the following game outputs 1 with negligible probability (in security parameter).

– *Initialize.* Run $(\mathsf{pp}, \mathsf{vk}, [\![\mathsf{sk}]\!]_n) \leftarrow \mathsf{Gen}(1^\lambda, n, t)$. Give pp, vk to $\mathcal{A}$. Receive the set of corrupt parties $C \subset [n]$ of size at most $t-1$ from $\mathcal{A}$. Then give $[\![\mathsf{sk}]\!]_C$ to $\mathcal{A}$. Define $\gamma := t - |C|$. Initiate a list $L := \emptyset$.
– *Signing queries.* On query (m, i) for $i \subseteq [n] \setminus C$ return $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m)$. Run this step as many times $\mathcal{A}$ desires.
– *Building the list.* If the number of signing query of the form (m, i) is at least γ, then insert m into the list L. (This captures that $\mathcal{A}$ has enough information to compute a signature on m.)
– *Output.* Eventually receive output $(m^\star, \sigma^\star)$ from $\mathcal{A}$. Return 1 if and only if $\mathsf{Ver}(\mathsf{vk}, (m^\star, \sigma^\star)) = 1$ and $m^\star \notin L$, and 0 otherwise.

4 Formalizing Fuzzy Threshold Tokenizer (FTT)

In this section we formally introduce the notion of *fuzzy threshold tokenizer* (FTT) and give a UC-secure definition. We first describe the algorithms/protocols in the primitive followed by the security definition in the next subsection.

Definition 4 (Fuzzy Threshold Tokenizer (FTT)). Given a security parameter $\lambda \in \mathbb{N}$, a threshold signature scheme $\mathsf{TS} = (\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$, biometric space parameters $q, \ell \in \mathbb{N}$, a distance predicate $\mathsf{Dist} : \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell \to \{0,1\}$, $n \in \mathbb{N}$ parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and a threshold of parties $t \in [n]$, a FTT scheme/protocol consists of the following tuple $(\mathsf{Setup}, \mathsf{Enrollment}, \mathsf{SignOn}, \mathsf{Ver})$ of algorithms/protocols:

– $\mathsf{Setup}(1^\lambda, n, t, \mathsf{TS}) \to (\mathsf{pp}_{\mathsf{setup}}, \{s_i, \mathsf{sk}_i^{\mathsf{TS}}\}_{i\in[n]}, \mathsf{vk})$: The Setup algorithm is run by a trusted authority. It first runs the key-generation of the threshold signature scheme, $(\{\mathsf{sk}_i^{\mathsf{TS}}\}_{i\in[n]}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda, n, t)$. It generates other public parameters $\mathsf{pp}_{\mathsf{setup}}$ and secret values $s_1, \ldots, s_n$ for each party respectively. It outputs $(\mathsf{vk}, \mathsf{pp}_{\mathsf{setup}})$ to every party and secrets $(\mathsf{sk}_i^{\mathsf{TS}}, s_i)$ to each party $\mathcal{P}_i$. ($\mathsf{pp}_{\mathsf{setup}}$ will be an implicit input in all the algorithms below.)
– $\mathsf{Enrollment}(n, t, q, \ell, \mathsf{Dist}) \to (\{a_i\}_{i\in[n]})$: On input the parameters from any party, this algorithm is run by the trusted authority to choose a random sample $\overrightarrow{\mathbf{w}} \leftarrow \mathcal{W}$. Then, each party $\mathcal{P}_i$ receives some information a_i.
– $\mathsf{SignOn}(\cdot)$: SignOn is a distributed protocol involving a party P^* along with a set S of parties. Party $\mathcal{P}^*$ has input a measurement $\overrightarrow{\mathbf{u}}$, message msg and its secret information $(s_*, \mathsf{sk}_*^{\mathsf{TS}})$. Each party $P_i \in S$ has input $(s_i, \mathsf{sk}_i^{\mathsf{TS}})$. At the end of the protocol, P^* obtains a (private) token Token (or $\perp$, denoting failure) as output. Each party $\mathcal{P}_i \in S$ gets output (msg, i, S). The trusted authority is not involved in this protocol.
– $\mathsf{Ver}(\mathsf{vk}, \mathsf{msg}, \mathsf{Token}) \to \{0,1\}$: Ver is an algorithm which takes input verification key vk, message msg and token Token, runs the verification algorithm of the threshold signature scheme $b := \mathsf{TS.Verify}(\mathsf{vk}, (\mathsf{msg}, \mathsf{Token}))$, and outputs $b \in \{0,1\}$. This can be run locally by any party or even any external entity.

Communication Model. In the $\mathsf{SignOn}(\cdot)$ protocol, only party $\mathcal{P}^*$ can communicate directly with every party in the set S. We stress that the other parties in S can not interact directly with each other.

4.1 Security Definition

We formally define security via the universal composability (UC) framework [26]. Similar to the simplified UC framework [28] we assume existence of *a default authenticated channel* in the real world. This simplifies the definition of our ideal functionality and can be removed easily by composing with an ideal authenticated channel functionality (e.g. [27]).

Consider n parties $P_1, \ldots, P_n$. We consider a *fixed number of parties* in the system throughout the paper. That is, no new party can join the execution subsequently. Let π^{TS} be an FTT scheme parameterized by a threshold signature scheme TS. Consider an adversarial environment $\mathcal{Z}$. We consider a *static corruption* model where there are a fixed set of corrupt parties decided a priori.[8] Informally, it is required that for every adversary $\mathcal{A}$ that corrupts some subset of the parties and participates in the real execution of the protocol, there exist an ideal world adversary Sim, such that for all environments $\mathcal{Z}$, the view of the environment is same in both worlds. We describe it more formally below.

Real world. In the real execution, the FTT protocol π^{TS} is executed in the presence of an adversary $\mathcal{A}$. The adversary $\mathcal{A}$ takes as input the security parameter λ and corrupts a subset of parties. Initially, the Setup algorithm is implemented by a trusted authority. The honest parties follow the instructions of π^{TS}. That is, whenever they receive an "Enrollment" query from $\mathcal{Z}$, they will run the $\mathsf{Enrollment}$ phase of π^{TS}. Similarly, whenever they receive a "Sign on" query from $\mathcal{Z}$ with input $(\mathsf{msg}, \overrightarrow{\mathbf{u}}, S)$, they will initiate a $\mathsf{SignOn}(\cdot)$ protocol with the parties in set S and using input $(\mathsf{msg}, S, \mathsf{sk}_i^{\mathsf{TS}})$. If a $\mathsf{SignOn}(\cdot)$ protocol is initiated with them by any other party, they participate honestly using input $\mathsf{sk}_i^{\mathsf{TS}}$. $\mathcal{A}$ sends all messages of the protocol on behalf of the corrupt parties following any arbitrary polynomial-time strategy. We assume that parties are connected by point to point secure and authenticated channels.

Ideal world. The ideal world is defined by a trusted ideal functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ described in Figure 1 that interacts with n (say) ideal dummy parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ and an ideal world adversary, a.k.a. the simulator Sim via secure (and authenticated) channels. The simulator can corrupt a subset of the parties and may fully control them. We discuss the ideal functionality in more detail later below.

The environment sets the inputs for all parties including the adversaries and obtain their outputs in both the worlds. However, the environment does *not* observe any internal interaction. For example, in the ideal world such interactions takes between the ideal functionality and another entity (a dummy party, or the simulator); in real world such interactions take place among the real parties. Finally, once the execution is over, the environment outputs a bit denoting either real or ideal world. For ideal functionality $\mathcal{F}$, adversary $\mathcal{A}$, simulator Sim,

[8] However, we allow the attacker to decide on the corrupt set adaptively after receiving the public values.

environment $\mathcal{Z}$ and a protocol π we formally denote the output of $\mathcal{Z}$ by random variable $\text{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ in the ideal world and $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ in the real world. We describe the ideal functionality for a FTT scheme in Figure 1 and we elaborate on it in the next subsection.

Definition 5 (UC-Realizing FTT). *Let* TS *be a threshold signature scheme (Definition 3),* $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ *be an ideal functionality as described in Figure 1 and* π^{TS} *be a FTT scheme.* π^{TS} *UC-realizes* $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ *if* for any *real world PPT adversary* $\mathcal{A}$*,* there exists *a PPT simulator* Sim *such that* for all *environments* $\mathcal{Z}$*,*

$$\text{IDEAL}_{\mathcal{F}_{\text{FTT}}^{\mathsf{TS}},\mathsf{Sim},\mathcal{Z}} \approx_c \text{REAL}_{\pi^{\mathsf{TS}},\mathcal{A},\mathcal{Z}}$$

Intuitively, for any adversary there should be a simulator that can simulate its behavior such that no environment can distinguish between these two worlds. Also, our definition can also capture setup assumptions such as random oracles by considering a $\mathcal{G}$-hybrid model with an ideal functionality $\mathcal{G}$ for the setup.

Ideal Functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ The ideal functionality we consider is presented formally in Figure 1. We provide an informal exposition here. Contrary to most of the UC ideal functionalities, our ideal functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ is parameterized with a threshold signature scheme $\mathsf{TS} = (\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ (see discussion about this choice later in this section). The ideal functionality is parameterized with a distance predicate Dist, which takes two vectors, a template and a candidate measurement and returns 1 if and only if the two vectors are "close". Additionally, the functionality is parameterized with other standard parameters and a probability distribution over the biometric vectors.

The ideal functionality has an interface to handle queries from different parties. For a particular session, the first query it responds to "`Setup`" from Sim. In response, the functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ generates the key pairs of the given threshold signature scheme, gives the control for the corrupt parties to the simulator and marks this session "LIVE". Then, an "`Enroll`" query can be made by any party. $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ chooses a template $\vec{\mathbf{w}}$ at random from the distribution $\mathcal{W}$, stores it and marks the session as "ENROLLED".

For any "ENROLLED" session, $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ can receive many "`SignOn`" queries (the previous two queries are allowed only once per session). This is ensured by not marking the session in response to any such query. The "`SignOn`" query from a party $\mathcal{P}_i$ contains a set S of parties (i.e. their identities), a message to be signed and a candidate measurement $\vec{\mathbf{u}}$. If the set S contains any corrupt party, $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ reaches out to the simulator for a response — this captures a corrupt party's power to deny a request.

Then, $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ checks whether the measurement $\vec{\mathbf{u}}$ is "close enough" by computing $b := \mathsf{Dist}(\vec{\mathbf{u}}, \vec{\mathbf{w}})$. If b is 1, the size of the set $S = t$ and all parties in S send an agreement response, $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ generates the partial signatures (tokens) on behalf of the parties in S and sends them *only to the initiator* $\mathcal{P}_i$; otherwise, it sends $\perp$ denoting failure to $\mathcal{P}_i$. Note that the signatures (or even the failure messages) are not sent to the simulator unless the initiator $\mathcal{P}_i$ is corrupt. This is crucial for our definition as it ensures that if a "`SignOn`" query is initiated by

Ideal Functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$

Given a threshold signature scheme ($\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify}$), the functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ is parameterized by a security parameter $\lambda \in \mathbb{N}$, biometric space parameters $q, \ell \in \mathbb{N}$, a distance predicate $\mathsf{Dist} : \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell \to \{0,1\}$, number of parties $n \in \mathbb{N}$ and a threshold of parties $t \in [n]$. It interacts with an ideal adversary (the simulator) Sim and n dummy parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ via the following queries.

- **On receiving a query of the form (**"`Setup`"**, sid, aux) from** Sim, do as follows only if sid is unmarked:
  1. run $(\mathsf{vk}, \{\mathsf{sk}_i^{\mathsf{TS}}\}_{i\in[n]}) \leftarrow \mathsf{TS.Gen}(1^\lambda)$;
  2. send ("`VerKey`", $\mathsf{sid}, \mathsf{vk}, \mathsf{aux}$) to Sim;
  3. receive ("`Corrupt`", $\mathsf{sid}, C \subseteq [n]$) from Sim;
  4. send $(\mathsf{sid}, \mathsf{sk}_i^{\mathsf{TS}})$ to each P_i for all $i \in [n]$.
  5. store the tuple $(\mathsf{sid}, \mathsf{vk}, \{\mathsf{sk}_i^{\mathsf{TS}}\}_{i\in[n]})$ and mark this session as "Live".
- **On receiving a query of the form (**"`Enroll`"**, sid) from** $\mathcal{P}$, only if sid is marked "Live":
  1. choose $\overrightarrow{\mathbf{w}} \leftarrow \mathcal{W}$ and store the tuple $(\mathsf{sid}, \overrightarrow{\mathbf{w}})$;
  2. send ("`Enrolled`", sid) to Sim and mark sid as "Enrolled".
- **On receiving a query of the form (**"`SignOn`"**,** $\mathsf{sid}, \mathsf{vk}, \mathsf{msg}, \mathcal{P}, \overrightarrow{\mathbf{u}}, S \subseteq [n]$**) from** $\mathcal{P}$, if the session sid is not marked "Enrolled", ignore this query. Else, retrieve the record $(\mathsf{sid}, \mathsf{pp}, \mathsf{vk}, \{\mathsf{sk}_i^{\mathsf{TS}}\}_{i\in[n]})$ and let $\{\mathcal{P}_j\}_{j\in S}$ be the parties in the set. Send (msg, P_i, S) to each P_j for $j \in S$. Then, if $S \cap C \neq \emptyset$ (contains a corrupt party), send ("`Signing Req`", $\mathsf{sid}, \mathsf{msg}, \mathcal{P}, S$) to Sim. If Sim sends back ("`Agreed`", sid, msg, $\mathcal{P}$) then do as follows:
  1. **if** $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$, $|S| = t$ then: generate $\{\mathsf{Token}_j \leftarrow \mathsf{TS.Sign}(\mathsf{sk}_j^{\mathsf{TS}}, \mathsf{msg})\}_{j\in S}$; and send $(\mathsf{sid}, \mathsf{msg}, \{\mathsf{Token}_j\}_{j\in S})$ to $\mathcal{P}$.
  2. **otherwise**, return $(\mathsf{sid}, \mathsf{msg}, \perp)$ to $\mathcal{P}$.

Fig. 1: The ideal functionality $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$.

an honest party, then the simulator does not obtain *anything* directly, except when there is a corrupt party in S via which it knows such a query has been made and only learns the tuple $(m, \mathcal{P}_i, S)$ corresponding to the query. In fact, no one except the initiator learns whether "`SignOn`" was successful. Intuitively, a protocol realizing $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$ must guarantee that a corrupt party can *not* compute a valid sign-on token (signature) just by participating in a session started by an honest party. In our definition of $\mathcal{F}_{\text{FTT}}^{\mathsf{TS}}$, such a token would be considered as a forgery. To the best of our knowledge, this feature has not been considered in prior works on threshold signatures.

We provide more discussions on our definition in the full version [12].

5 Any Distance Measure from MPC

In this section, we show how to construct a four round secure fuzzy threshold tokenizer using any two round malicious UC-secure MPC protocol in a broadcast

channel as the main technical tool. Our tokenizer scheme satisfies Definition 1 for any n, t, for any distance measure. Formally, we show the following theorem:

Theorem 1. *Assuming unforgeable threshold signatures and a two round UC-secure MPC protocols in the CRS model in a broadcast channel, there exists a four round secure fuzzy threshold tokenizer protocol for any n, t and any distance predicate.*

Such two round MPC protocols can be built assuming DDH/LWE/QR/N^{th} Residuosity [46,50,38,15]. Threshold signatures can be built assuming LWE/Gap-DDH/RSA [20,18,55]. Instantiating this, we get the following corollary:

Corollary 1. *Assuming LWE, there exists a four round secure FTT protocol for any n, t and any distance predicate.*

We describe the construction below and defer the proof to the full version [12].

5.1 Construction

Notation. Let π be a two round UC-secure MPC protocol in the CRS model in the presence of a broadcast channel that is secure against a malicious adversary that can corrupt upto $(t-1)$ parties. Let $\pi.\mathsf{Setup}$ denote the algorithm used to generate the CRS. Let $(\pi.\mathsf{Round}_1, \pi.\mathsf{Round}_2)$ denote the algorithms used by any party to compute the messages in each of the two rounds and $\pi.\mathsf{Out}$ denote the algorithm to compute the final output. Let $(\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ be a threshold signature scheme, $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ be a secret key encryption scheme, $(\mathsf{Share}, \mathsf{Recon})$ be a (t, n) threshold secret sharing scheme and PRF be a pseudorandom function. We now describe the construction of our four round secure fuzzy threshold tokenizer protocol π^{Any} for any n and t.

Setup: The following algorithm is executed by a trusted authority:

- Generate $\mathsf{crs} \leftarrow \pi.\mathsf{Setup}(1^\lambda)$.
- For each $i \in [n]$, compute $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$.
- For every $i, j \in [n]$, compute $(\mathsf{k}^{\mathsf{PRF}}_{i,j}, \mathsf{k}^{\mathsf{PRF}}_{j,i})$ as uniformly random strings.
- Compute $(\mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}^{\mathsf{TS}}_1, \ldots, \mathsf{sk}^{\mathsf{TS}}_n) \leftarrow \mathsf{TS.Gen}(1^\lambda, n, t)$.
- For each $i \in [n]$, give $(\mathsf{crs}, \mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}^{\mathsf{TS}}_i, \mathsf{sk}_i, \{\mathsf{vk}_j\}_{j\in[n]}, \{\mathsf{k}^{\mathsf{PRF}}_{j,i}, \mathsf{k}^{\mathsf{PRF}}_{i,j}\}_{j\in[n]})$ to party $\mathcal{P}_i$.

Enrollment: In this phase, any party $\mathcal{P}_i$ that wishes to enroll queries the trusted authority which then does the following:

- Sample a random vector $\overrightarrow{\mathbf{w}}$ from the distribution $\mathcal{W}$.
- Compute $(\overrightarrow{\mathbf{w}}_1, \ldots, \overrightarrow{\mathbf{w}}_n) \leftarrow \mathsf{Share}(1^\lambda, \overrightarrow{\mathbf{w}}, n, t)$.
- For each $i \in [n]$, give $(\overrightarrow{\mathbf{w}}_i)$ to party $\mathcal{P}_i$.

SignOn Phase: In the SignOn phase, let's consider party $\mathcal{P}^*$ that uses input vector $\overrightarrow{\mathbf{u}}$, a message msg on which it wants a token. $\mathcal{P}^*$ interacts with the other parties in the below four round protocol.

Round 1: $(\mathcal{P}^* \rightarrow)$ [9] Party $\mathcal{P}^*$ does the following:

[9] The arrowhead denotes that in this round messages are outgoing from party $\mathcal{P}^*$.

1. Pick a set S consisting of t parties amongst $\mathcal{P}_1, \ldots, \mathcal{P}_n$. For simplicity, without loss of generality, we assume that $\mathcal{P}^*$ is also part of set S.
2. To each party $\mathcal{P}_i \in S$, send (msg, S).

Round 2: $(\rightarrow \mathcal{P}^*)$ Each Party $\mathcal{P}_i \in S$ (except $\mathcal{P}^*$) does the following:

1. Participate in an execution of protocol π with parties in set S using input $\mathsf{y}_i = (\overrightarrow{\mathbf{w}}_i, \mathsf{sk}_i^{\mathsf{TS}})$ and randomness r_i to compute circuit $\mathcal{C}$ defined in Figure 2. Compute first round message $\mathsf{msg}_{1,i} \leftarrow \pi.\mathsf{Round}_1(\mathsf{y}_i; \mathsf{r}_i)$.
2. Compute $\sigma_{1,i} = \mathsf{Sign}(\mathsf{sk}_i, \mathsf{msg}_{1,i})$.
3. Send $(\mathsf{msg}_{1,i}, \sigma_{1,i})$ to party $\mathcal{P}^*$.

Round 3: $(\mathcal{P}^* \rightarrow)$ Party $\mathcal{P}^*$ does the following:

1. Let $\mathsf{Trans}_{\text{fuzzy threshold tokenizer}}$ denote the set of messages received in round 2.
2. Participate in an execution of protocol π with parties in set S using input $\mathsf{y}_* = (\overrightarrow{\mathbf{w}}_*, \mathsf{sk}_*^{\mathsf{TS}}, \overrightarrow{\mathbf{u}}, \mathsf{msg})$ and randomness r_* to compute circuit $\mathcal{C}$ defined in Figure 2. Compute first round message $\mathsf{msg}_{1,*} \leftarrow \pi.\mathsf{Round}_1(\mathsf{y}_*; \mathsf{r}_*)$.
3. To each party $\mathcal{P}_i \in S$, send $(\mathsf{Trans}_{\text{fuzzy threshold tokenizer}}, \mathsf{msg}_{1,*})$.

Round 4: $(\rightarrow \mathcal{P}^*)$ Each Party $\mathcal{P}_i \in S$ (except $\mathcal{P}^*$) does the following:

1. Let $\mathsf{Trans}_{\text{fuzzy threshold tokenizer}}$ consist of a set of messages of the form $(\mathsf{msg}_{1,j}, \sigma_{1,j})$, $\forall j \in S \setminus \mathcal{P}^*$. Output $\perp$ if $\mathsf{Verify}(\mathsf{vk}_j, \mathsf{msg}_{1,j}, \sigma_{1,j}) \neq 1$.
2. Let $\tau_1 = \{\mathsf{msg}_{1,j}\}_{j \in S}$ denote the transcript of protocol π after round 1. Compute second round message $\mathsf{msg}_{2,i} \leftarrow \pi.\mathsf{Round}_2(\mathsf{y}_i, \tau_1; \mathsf{r}_i)$.
3. Let $(\mathsf{Trans}_{\text{fuzzy threshold tokenizer}}, \mathsf{msg}_{1,*})$ denote the message received from $\mathcal{P}^*$ in round 3. Compute $\mathsf{ek}_i = \oplus_{j \in S} \mathsf{PRF}(\mathsf{k}_{i,j}^{\mathsf{PRF}}, \mathsf{msg}_{1,*})$ and $\mathsf{ct}_i = \mathsf{SKE.Enc}(\mathsf{ek}_i, \mathsf{msg}_{2,i})$.
4. For each party $\mathcal{P}_j \in S$, compute $\mathsf{ek}_{j,i} = \mathsf{PRF}(\mathsf{k}_{j,i}^{\mathsf{PRF}}, \mathsf{msg}_{1,*})$.
5. Send $(\mathsf{ct}_i, \{\mathsf{ek}_{j,i}\}_{j \in S})$ to $\mathcal{P}^*$.

Output Computation: Every party $\mathcal{P}_j \in S$ outputs $(\mathsf{msg}, \mathcal{P}^*, S)$. Additionally, party $\mathcal{P}^*$ does the following to generate a token:

1. For each party $\mathcal{P}_j \in S$, compute $\mathsf{ek}_j = \oplus_{j \in S} \mathsf{ek}_{j,i}$, $\mathsf{msg}_{2,j} = \mathsf{SKE.Dec}(\mathsf{ek}_j, \mathsf{ct}_j)$.
2. Let τ_2 denote the transcript of protocol π after round 2. Compute the output of π: $\{\mathsf{Token}_i\}_{i \in S} \leftarrow \pi.\mathsf{Out}(\mathsf{y}_*, \tau_2; \mathsf{r}_*)$.
3. Reconstruct the signature as $\mathsf{Token} = \mathsf{TS.Combine}(\{\mathsf{Token}_i\}_{i \in S})$.
4. If $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token}) = 1$, then output $\{\mathsf{Token}_i\}_{i \in S}$. Else, output $\perp$.

Token Verification: Given a verification key $\mathsf{vk}^{\mathsf{TS}}$, message msg and a token $\{\mathsf{Token}_i\}_{i \in S}$, where $|S| = t$, the token verification algorithm does the following:

1. Compute $\mathsf{Token} \leftarrow \mathsf{TS.Combine}(\{\mathsf{Token}_i\}_{i \in S})$.
2. Output 1 if $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token}) = 1$. Else, output 0.

Inputs:

- Party $\mathcal{P}_i \in S$ has input $(\vec{\mathbf{w}}_i, \mathsf{sk}_i^{\mathsf{TS}})$.
- Party $\mathcal{P}^* \in S$ additionally has input $(\vec{\mathbf{u}}, \mathsf{msg})$.

Computation:

- Compute $\vec{\mathbf{w}} = \mathsf{Recon}(\{\vec{\mathbf{w}}_i\}_{i \in S})$. Output $\perp$ to $\mathcal{P}^*$ if the reconstruction fails or if $\mathsf{Dist}(\vec{\mathbf{u}}, \vec{\mathbf{w}}) = 0$.
- Compute $\mathsf{Token}_i = \mathsf{TS.Sign}(\mathsf{sk}_i^{\mathsf{TS}}, \mathsf{msg})$. Output $\{\mathsf{Token}_i\}_{i \in S}$ to party $\mathcal{P}^*$.

Fig. 2: Circuit $\mathcal{C}$

6 Any Distance Measure using Threshold FHE

In this section, we construct a FTT protocol for any distance measure using any fully homomorphic encryption (FHE) scheme with threshold decryption. Our token generation protocol satisfies the definition in Section 4 for any n, t, and works for any distance measure. Formally, we show the following theorem:

Theorem 2. *Assuming threshold fully-homomorphic encryption, non-interactive zero knowledge argument of knowledge (NIZK) and unforgeable threshold signatures, there exists a four round secure FTT protocol for any n, t and any distance predicate.*

Threshold FHE, NIZKs and unforgeable threshold signatures can be built assuming LWE [20,51]. Instantiating this, we get the following corollary:

Corollary 2. *Assuming LWE, there exists a four round secure FTT protocol for any n, t and any distance predicate.*

6.1 Construction

Notation. Let $(\mathsf{TFHE.Gen}, \mathsf{TFHE.Enc}, \mathsf{TFHE.PartialDec}, \mathsf{TFHE.Eval}, \mathsf{TFHE.Combine})$ be a threshold FHE scheme and let $(\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ be a threshold signature scheme. Let $(\mathsf{Prove}, \mathsf{Verify})$ be a NIZK scheme and $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a strongly-unforgeable digital signature scheme and Commit be a non-interactive commitment scheme. We now describe the construction of our four round secure FTT protocol $\pi^{\mathsf{Any-TFHE}}$ for any n and k. We defer the proof to the full version [12].

Setup Phase: The following algorithm is executed by a trusted authority:

- Generate $(\mathsf{pk}^{\mathsf{TFHE}}, \mathsf{sk}_1^{\mathsf{TFHE}}, \ldots, \mathsf{sk}_N^{\mathsf{TFHE}}) \leftarrow \mathsf{TFHE.Gen}(1^\lambda, n, t)$ and $(\mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}_1^{\mathsf{TS}}, \ldots, \mathsf{sk}_n^{\mathsf{TS}}) \leftarrow \mathsf{TS.Gen}(1^\lambda, n, t)$.
- For each $i \in [n]$, compute $\mathsf{com}_i \leftarrow \mathsf{Commit}(\mathsf{sk}_i^{\mathsf{TFHE}}; \mathsf{r}_i^{\mathsf{com}})$ and $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{Gen}(1^\lambda)$.
- For each $i \in [n]$, give the following to party $\mathcal{P}_i$: $(\mathsf{pk}^{\mathsf{TFHE}}, \mathsf{sk}_i^{\mathsf{TFHE}}, \mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}_i^{\mathsf{TS}}, (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), \mathsf{sk}_i, (\mathsf{com}_1, \ldots, \mathsf{com}_n), \mathsf{r}_i^{\mathsf{com}})$.

Enrollment: In this phase, any party $\mathcal{P}_i$ that wishes to register a fresh template queries the trusted authority, which then executes the following algorithm:

- Sample a template $\overrightarrow{\mathbf{w}}$ from the distribution $\mathcal{W}$ over $\{0,1\}^\ell$.
- Compute and give $\mathsf{ct}_{\overrightarrow{\mathbf{w}}}$ to each party $\mathcal{P}_i$, where $\mathsf{ct}_{\overrightarrow{\mathbf{w}}} = \mathsf{TFHE.Enc}(\mathsf{pk}^{\mathsf{TFHE}}, \overrightarrow{\mathbf{w}})$.

SignOn Phase: In the SignOn phase, let's consider party $\mathcal{P}^*$ that uses input vector $\overrightarrow{\mathbf{u}} \in \{0,1\}^\ell$ and a message msg on which it wants a token. $\mathcal{P}^*$ interacts with the other parties in the below four round protocol.

- **Round 1:** $(\mathcal{P}^* \rightarrow)$ [10] Party $\mathcal{P}^*$ does the following:
  1. Compute ciphertext $\mathsf{ct}_{\overrightarrow{\mathbf{u}}} = \mathsf{TFHE.Enc}(\mathsf{pk}^{\mathsf{TFHE}}, \overrightarrow{\mathbf{u}}; \mathsf{r}_{\overrightarrow{\mathbf{u}}})$.
  2. Compute $\pi_{\overrightarrow{\mathbf{u}}} \leftarrow \mathsf{Prove}(\mathsf{st}_{\overrightarrow{\mathbf{u}}}, \mathsf{wit}_{\overrightarrow{\mathbf{u}}})$ for $\mathsf{st}_{\overrightarrow{\mathbf{u}}} = (\mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \mathsf{pk}^{\mathsf{TFHE}}) \in L_1$ using witness $\mathsf{wit}_{\overrightarrow{\mathbf{u}}} = (\overrightarrow{\mathbf{u}}, \mathsf{r}_{\overrightarrow{\mathbf{u}}})$ (language L_1 is defined in Figure 3).
  3. Pick a set S consisting of t parties amongst $\mathcal{P}_1, \ldots, \mathcal{P}_n$. For simplicity, without loss of generality, we assume that $\mathcal{P}^*$ is also part of set S.
  4. To each party $\mathcal{P}_i \in S$, send $(\mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \pi_{\overrightarrow{\mathbf{u}}})$.

Statement: The statement st is as follows: $\mathsf{st} = (\mathsf{ct}, \mathsf{pk})$.
Witness: The witness wit is as follows: $\mathsf{wit} = (\mathsf{x}, \mathsf{r})$.
Relation: $\mathsf{R}_1(\mathsf{st}, \mathsf{wit}) = 1$ if and only if $\mathsf{ct} = \mathsf{TFHE.Enc}(\mathsf{pk}, \mathsf{x}; \mathsf{r})$.

Fig. 3: NP language L_1

- **Round 2:** $(\rightarrow \mathcal{P}^*)$ Each party $\mathcal{P}_i \in S$ (except $\mathcal{P}^*$) does the following:
  1. Abort and output $\perp$ if $\mathsf{Verify}(\pi_{\overrightarrow{\mathbf{u}}}, \mathsf{st}_{\overrightarrow{\mathbf{u}}}) \neq 1$ for language L_1 where the statement $\mathsf{st}_{\overrightarrow{\mathbf{u}}} = (\mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \mathsf{pk}^{\mathsf{TFHE}})$.
  2. Sample a uniformly random one-time key $K_i \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{ct}_{K_i} = \mathsf{TFHE.Enc}\ (\mathsf{pk}^{\mathsf{TFHE}}, K_i; \mathsf{r}_{K_i})$.
  3. Compute $\pi_{K_i} \leftarrow \mathsf{Prove}(\mathsf{st}_{K_i}, \mathsf{wit}_{K_i})$ for $\mathsf{st}_{K_i} = (\mathsf{ct}_{K_i}, \mathsf{pk}^{\mathsf{TFHE}}) \in L_1$ using the witness $\mathsf{wit}_{K_i} = (K_i, \mathsf{r}_{K_i})$ (language L_1 is defined in Figure 3).
  4. Compute signatures $\sigma_{i,0} = \mathsf{Sign}(\mathsf{sk}_i, \mathsf{ct}_{\overrightarrow{\mathbf{u}}})$ and $\sigma_{i,1} = \mathsf{Sign}(\mathsf{sk}_i, \mathsf{ct}_{K_i})$.
  5. Send the following to the party $\mathcal{P}^*$: $(\mathsf{ct}_{K_i}, \pi_{K_i}, \sigma_{i,0}, \sigma_{i,1})$.
- **Round 3:** $(\mathcal{P}^* \rightarrow)$ Party $\mathcal{P}^*$ checks if there exists some party $\mathcal{P}_i \in S$ such that $\mathsf{Verify}(\pi_{K_i}, \mathsf{st}_{K_i}) \neq 1$ for language L_1 where $\mathsf{st}_{K_i} = (\mathsf{ct}_{K_i}, \mathsf{pk}^{\mathsf{TFHE}})$. If yes, it outputs $\perp$ and aborts. Otherwise, it sends $\{(\mathsf{ct}_{K_i}, \pi_{K_i}, \sigma_{i,0}, \sigma_{i,1})\}_{\mathcal{P}_i \in S}$ to each party $\mathcal{P}_i \in S$.
- **Round 4:** $(\rightarrow \mathcal{P}^*)$ Each party $\mathcal{P}_i \in S$ (except $\mathcal{P}^*$) does the following:

[10] The arrowhead denotes that in this round messages are outgoing from party $\mathcal{P}^*$.

1. If there exists some party $\mathcal{P}_j \in S$ such that $\mathsf{Verify}(\pi_{K_j}, \mathsf{st}_{K_j}) \neq 1$ for language L_1 where $\mathsf{st}_{K_j} = (\mathsf{ct}_{K_j}, \mathsf{pk}^{\mathsf{TFHE}})$ (OR) $\mathsf{Verify}(\mathsf{vk}_j, \mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \sigma_{j,0}) \neq 1$ (OR) $\mathsf{Verify}(\mathsf{vk}_j, \mathsf{ct}_{K_j}, \sigma_{j,1}) \neq 1$, then output $\perp$ and abort.
2. Otherwise, for each $\mathcal{P}_j \in S$, do the following:
   - Compute $\mathsf{ct}_{\mathcal{C},j} = \mathsf{TFHE.Eval}(\mathsf{pk}^{\mathsf{TFHE}}, \mathcal{C}_{\mathsf{Dist}}, \mathsf{ct}_{\overrightarrow{\mathbf{w}}}, \mathsf{ct}_{\overrightarrow{\mathbf{u}}}, \mathsf{ct}_{K_j})$ using circuit $\mathcal{C}$ (Figure 4). Note that $\mathsf{ct}_{\mathcal{C},j}$ is either an encryption K_j or an encryption of 0^λ.

Inputs: A template $\overrightarrow{\mathbf{w}} \in \{0,1\}^\ell$, measurement $\overrightarrow{\mathbf{u}} \in \{0,1\}^\ell$ and string $K \in \{0,1\}^\lambda$.

Computation: If $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$, output K. Else, output 0^λ.

Fig. 4: Circuit $\mathcal{C}$

   - Compute a partial decryption: $\mu_{i,j} = \mathsf{TFHE.PartialDec}(\mathsf{sk}_i^{\mathsf{TFHE}}, \mathsf{ct}_{\mathcal{C},j})$.
   - Compute $\pi_{i,j} \leftarrow \mathsf{Prove}(\mathsf{st}_{i,j}, \mathsf{wit}_i)$ for $\mathsf{st}_{i,j} = (\mathsf{ct}_{\mathcal{C},j}, \mu_{i,j}, \mathsf{com}_i) \in L_2$ using $\mathsf{wit}_i = (\mathsf{sk}_i^{\mathsf{TFHE}}, \mathsf{r}_i^{\mathsf{com}})$ (language L_2 is defined in Figure 5).
3. Compute partial signature $\mathsf{Token}_i = \mathsf{TS.Sign}(\mathsf{sk}_i^{\mathsf{TFHE}}, \mathsf{msg})$ and ciphertext $\mathsf{ct}_i = K_i \oplus \mathsf{Token}_i$.
4. Send $(\mathsf{ct}_i, \{(\pi_{i,j}, \mu_{i,j})\}_{\mathcal{P}_j \in S})$ to $\mathcal{P}^*$.

Statement: The statement st is as follows: $\mathsf{st} = (\mathsf{ct}, \mu, \mathsf{com})$.

Witness: The witness wit is as follows: $\mathsf{wit} = (\mathsf{sk}^{\mathsf{TFHE}}, \mathsf{r})$.

Relation: $\mathsf{R}_2(\mathsf{st}, \mathsf{wit}) = 1$ if and only if: (a) $\mathsf{TFHE.PartialDec}(\mathsf{sk}^{\mathsf{TFHE}}, \mathsf{ct}) = \mu$ *and* (b) $\mathsf{Commit}(\mathsf{sk}^{\mathsf{TFHE}}, \mathsf{r}) = \mathsf{com}$.

Fig. 5: NP language L_2

– **Output Computation:** Every party $\mathcal{P}_i \in S$ outputs $(\mathsf{msg}, \mathcal{P}^*, S)$. Additionally, party $\mathcal{P}^*$ does the following to generate a token:
1. For each $\mathcal{P}_j \in S$, do the following:
   (a) For each $\mathcal{P}_i \in S$, abort if $\mathsf{Verify}(\pi_{i,j}, \mathsf{st}_{i,j}) \neq 1$ for language L_2 where $\mathsf{st}_{i,j} = (\mathsf{ct}_{\mathcal{C},j}, \mu_{i,j}, \mathsf{com}_i)$.
   (b) Set $K_j = \mathsf{TFHE.Combine}(\{\mu_{i,j}\}_{\mathcal{P}_i \in S})$. If $K_j = 0^\lambda$, output $\perp$.
   (c) Otherwise, recover partial signature $\mathsf{Token}_j = K_j \oplus \mathsf{ct}_j$.
2. Reconstruct the signature as $\mathsf{Token} = \mathsf{TS.Combine}(\{\mathsf{Token}_i\}_{i \in S})$.
3. If $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token}) = 1$, then output $\{\mathsf{Token}_i\}_{\mathcal{P}_i \in S}$. Else, output $\perp$.

Token Verification: Given a verification key $\mathsf{vk}^{\mathsf{TS}}$, message msg and a set of partial tokens $\{\mathsf{Token}_i\}_{\mathcal{P}_i \in S}$, the token verification algorithm outputs 1 if $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token}) = 1$, where $\mathsf{Token} = \mathsf{TS.Combine}(\{\mathsf{Token}_i\}_{\mathcal{P}_i \in S})$.

7 Cosine Similarity: Single Corruption

In this section, we construct an efficient four round secure FTT in the Random Oracle (RO) model for Euclidean Distance and Cosine Similarity. Our protocol satisfies Definition 1 for any n with threshold $t = 3$ and is secure against a malicious adversary that can corrupt any one party. The special case of $n = 3$ corresponds to the popularly studied three party honest majority setting. We first focus on the Cosine Similarity distance measure. In the full version, we explain how to extend our result for Euclidean Distance. Formally:

Theorem 3. *Assuming unforgeable threshold signatures, two message OT in the CRS model, circuit-private additively homomorphic encryption and NIZKs for NP languages L_1, L_2 defined below, there exists a four round secure fuzzy threshold tokenizer protocol for Cosine Similarity. The protocol works for any n, threshold $t = 3$ and is secure against a malicious adversary that can corrupt any one party.*

We describe the construction below and defer the proof to the full version [12].

Paillier Encryption Scheme. The Paillier encryption scheme [48] is an example of a circuit-private additively homomorphic encryption based on the N^{th} residuosity assumption. With respect to Paillier, we can also build NIZK arguments for languages L_1 and L_2 defined below, in the RO model. Formally:

Imported Theorem 1 ([31]) *Assuming the hardness of the N^{th} residuosity assumption, there exists a NIZK for language L_1, defined below, in the RO model.*

Imported Theorem 2 ([30]) *Assuming the hardness of the N^{th} residuosity assumption, there exists a NIZK for language L_2, defined below, in the RO model.*

The above NIZKs are very efficient and only require a constant number of group operations for both prover and verifier. Two message OT in the CRS model can be built assuming DDH/LWE/Quadratic Residuosity/N^{th} residuosity [47,52,40]. Threshold signatures can be built assuming LWE/Gap-DDH/RSA [20,18,55]. Instantiating the primitives used in Theorem 3, we get the following corollary:

Corollary 3. *Assuming the hardness of the N^{th} residuosity assumption and LWE, there exists a four round secure fuzzy threshold tokenizer protocol for Cosine Similarity in the RO model. The protocol works for any n, $t = 3$ and is secure against a malicious adversary that can corrupt any one party.*

NP Languages.
Let $(\mathsf{AHE.Setup}, \mathsf{AHE.Enc}, \mathsf{AHE.Add}, \mathsf{AHE.ConstMul}, \mathsf{AHE.Dec})$ be an additively homomorphic encryption scheme. Let $\mathsf{epk} \leftarrow \mathsf{AHE.Setup}(1^\lambda)$, $m = \mathsf{poly}(\lambda)$.
Language L_1:
Statement: $\mathsf{st} = (\mathsf{ct}, \mathsf{pk})$. **Witness:** $\mathsf{wit} = (\mathsf{x}, \mathsf{r})$.
Relation: $\mathsf{R}_1(\mathsf{st}, \mathsf{wit}) = 1$ if $\mathsf{ct} = \mathsf{AHE.Enc}(\mathsf{epk}, \mathsf{x}; \mathsf{r})$ AND $x \in \{0,1\}^m$

Language L_2:
Statement: $\mathsf{st} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{pk})$. **Witness:** $\mathsf{wit} = (\mathsf{x}_2, \mathsf{r}_2, \mathsf{r}_3)$.
Relation: $\mathsf{R}_2(\mathsf{st}, \mathsf{wit}) = 1$ if

$$\mathsf{ct}_2 = \mathsf{AHE.Enc}(\mathsf{epk}, \mathsf{x}_2; \mathsf{r}_2) \text{ AND } \mathsf{ct}_3 = \mathsf{AHE.ConstMul}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{x}_2; \mathsf{r}_3).$$

Construction. Let RO denote a random oracle, d be the threshold value for Cosine Similarity. Recall that we denote $\mathsf{Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) = 1$ if $\mathsf{CS.Dist}(\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}) \geq d$. Let $(\mathsf{Share}, \mathsf{Recon})$ be a $(2, n)$ threshold secret sharing scheme, $\mathsf{TS} = (\mathsf{TS.Gen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Verify})$ be a threshold signature scheme, $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ denote a secret key encryption scheme, PRF denote a pseudorandom function, $(\mathsf{Garble}, \mathsf{Eval})$ denote a garbling scheme for circuits, $(\mathsf{Prove}, \mathsf{Verify})$ be a NIZK system in the RO model, $\mathsf{AHE} = (\mathsf{AHE.Setup}, \mathsf{AHE.Enc}, \mathsf{AHE.Add}, \mathsf{AHE.ConstMul}, \mathsf{AHE.Dec})$ be a circuit-private additively homomorphic encryption scheme and $\mathsf{OT} = (\mathsf{OT.Setup}, \mathsf{OT.Round}_1, \mathsf{OT.Round}_2, \mathsf{OT.Output})$ be a two message oblivious transfer protocol in the CRS model. We now describe the construction of our four round secure fuzzy threshold tokenizer protocol π^{CS} for Cosine Similarity.

Setup: The trusted authority does the following:

- Compute $(\mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}_1^{\mathsf{TS}}, \ldots, \mathsf{sk}_n^{\mathsf{TS}}) \leftarrow \mathsf{TS.Gen}(1^\lambda, n, k)$.
- For $i \in [n]$, generate $\mathsf{crs}_i \leftarrow \mathsf{OT.Setup}(1^\lambda)$ and pick a random PRF key k_i.
- For $i \in [n]$, give $(\mathsf{pp}^{\mathsf{TS}}, \mathsf{vk}^{\mathsf{TS}}, \mathsf{sk}_i^{\mathsf{TS}}, \{\mathsf{crs}_j\}_{j\in[n]}, \{\mathsf{k}_j\}_{j\in[n]\setminus i})$ to party $\mathcal{P}_i$.

Enrollment: In this phase, any party $\mathcal{P}_i$ that wishes to enroll, queries the trusted authority which then does the following:

- Sample a random vector $\overrightarrow{\mathbf{w}}$ from the distribution $\mathcal{W}$. Without loss of generality, let's assume that the $\mathsf{L2}$-norm of $\overrightarrow{\mathbf{w}}$ is 1.
- For each $i \in [n]$, do the following:
  - Compute $(\overrightarrow{\mathbf{w}}_i, \overrightarrow{\mathbf{v}}_i) \leftarrow \mathsf{Share}(1^\lambda, \overrightarrow{\mathbf{w}}, n, 2)$.
  - Compute $(\mathsf{esk}_i, \mathsf{epk}_i) \leftarrow \mathsf{AHE.Setup}(1^\lambda)$.
  - Let $\overrightarrow{\mathbf{w}}_i = (\mathsf{w}_{i,1}, \ldots, \mathsf{w}_{i,\ell})$. $\forall j \in [\ell]$, compute $[\![\mathsf{w}_{i,j}]\!] = \mathsf{AHE.Enc}(\mathsf{epk}_i, \mathsf{w}_{i,j})$.
  - Give $(\overrightarrow{\mathbf{w}}_i, \mathsf{sk}_i, \mathsf{pk}_i, \{[\![\mathsf{w}_{i,j}]\!]\}_{j\in[\ell]})$ to party $\mathcal{P}_i$ and $(\overrightarrow{\mathbf{v}}_i, \mathsf{pk}_i, \{[\![\mathsf{w}_{i,j}]\!]\}_{j\in[\ell]})$ to all the other parties.

SignOn Phase: In the SignOn phase, let's consider party $\mathcal{P}_i$ that uses an input vector $\overrightarrow{\mathbf{u}} = (\mathsf{u}_1, \ldots, \mathsf{u}_\ell)$ and a message msg on which it wants a token. $\mathcal{P}_i$ picks two other parties $\mathcal{P}_j$ and $\mathcal{P}_k$ and interacts with them in the below protocol.

Round 1: $(\mathcal{P}_i \rightarrow)$ [11] Party $\mathcal{P}_i$ does the following:

1. Let $S = (\mathcal{P}_j, \mathcal{P}_k)$ with $j < k$.
2. For each $j \in [\ell]$, compute the following:
   - $[\![\mathsf{u}_j]\!] = \mathsf{AHE.Enc}(\mathsf{epk}_i, \mathsf{u}_j; \mathsf{r}_{1,j})$. $\pi_{1,j} \leftarrow \mathsf{Prove}(\mathsf{st}_{1,j}, \mathsf{wit}_{1,j})$ for $\mathsf{st}_{1,j} = ([\![\mathsf{u}_j]\!], \mathsf{epk}_i) \in L_1$ using $\mathsf{wit}_{1,j} = (\mathsf{u}_j, \mathsf{r}_{1,j})$.

[11] The arrowhead denotes that in this round messages are outgoing from party $\mathcal{P}_i$.

- $[\![\mathsf{u}_j^2]\!] = \mathsf{AHE.ConstMul}(\mathsf{epk}_i, [\![\mathsf{u}_j]\!], \mathsf{u}_j; \mathsf{r}_{2,j})$. $\pi_{2,j} \leftarrow \mathsf{Prove}(\mathsf{st}_{2,j}, \mathsf{wit}_{2,j})$ for $\mathsf{st}_{2,j} = ([\![\mathsf{u}_j]\!], [\![\mathsf{u}_j]\!], [\![\mathsf{u}_j^2]\!], \mathsf{epk}_i) \in L_2$ using $\mathsf{wit}_{2,j} = (\mathsf{u}_j, \mathsf{r}_{1,j}, \mathsf{r}_{2,j})$.
- $[\![\mathsf{w}_{i,j} \cdot \mathsf{u}_j]\!] = \mathsf{AHE.ConstMul}(\mathsf{epk}_i, [\![\mathsf{w}_{i,j}]\!], \mathsf{u}_j; \mathsf{r}_{3,j})$. $\pi_{3,j} \leftarrow \mathsf{Prove}(\mathsf{st}_{3,j}, \mathsf{wit}_{3,j})$ for $\mathsf{st}_{3,j} = ([\![\mathsf{w}_{i,j}]\!], [\![\mathsf{u}_j]\!], [\![\mathsf{w}_{i,j} \cdot \mathsf{u}_j]\!], \mathsf{epk}_i) \in L_2$ using $\mathsf{wit}_{3,j} = (\mathsf{u}_j, \mathsf{r}_{1,j}, \mathsf{r}_{3,j})$.

3. To both parties in S, send $\mathsf{msg}_1 = (S, \mathsf{msg}, \{[\![\mathsf{u}_j]\!], [\![\mathsf{u}_j^2]\!], [\![\mathsf{w}_{i,j} \cdot \mathsf{u}_j]\!], \pi_{1,j}, \pi_{2,j}, \pi_{3,j}\}_{j \in [\ell]})$.

Round 2: $(\rightarrow \mathcal{P}_i)$ Both parties $\mathcal{P}_j$ and $\mathcal{P}_k$ do the following:

1. Abort if any of the proofs $\{\pi_{1,j}, \pi_{2,j}, \pi_{3,j}\}_{j \in [\ell]}$ don't verify.
2. Generate randomness $(\mathsf{a}, \mathsf{b}, \mathsf{e}, \mathsf{f}, \mathsf{p}, \mathsf{q}, \mathsf{r}_\mathsf{z}) \leftarrow \mathsf{PRF}(\mathsf{k}_i, \mathsf{msg}_1)$.
3. Using the algorithms of AHE, compute $[\![\mathsf{x}_1]\!], [\![\mathsf{x}_2]\!], [\![\mathsf{y}_1]\!], [\![\mathsf{y}_2]\!], [\![\mathsf{z}_1]\!], [\![\mathsf{z}_2]\!]$ as follows:
   - $\mathsf{x}_1 = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_i \rangle$, $\mathsf{y}_1 = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle$, $\mathsf{z}_1 = (\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{v}}_i \rangle + \mathsf{r}_\mathsf{z})$.
   - $\mathsf{x}_2 = (\mathsf{a} \cdot \mathsf{x}_1 + \mathsf{b})$, $\mathsf{y}_2 = (\mathsf{e} \cdot \mathsf{y}_1 + \mathsf{f})$, $\mathsf{z}_2 = (\mathsf{p} \cdot \mathsf{z}_1 + \mathsf{q})$
4. Send $([\![\mathsf{x}_2]\!], [\![\mathsf{y}_2]\!], [\![\mathsf{z}_1]\!], [\![\mathsf{z}_2]\!])$ to $\mathcal{P}_i$.

Round 3: $(\mathcal{P}_i \rightarrow)$ Party $\mathcal{P}_i$ does the following:

1. Abort if the tuples sent by both $\mathcal{P}_j$ and $\mathcal{P}_k$ in round 2 were not the same.
2. Compute $\mathsf{x}_1 = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{w}}_i \rangle$, $\mathsf{x}_2 = \mathsf{AHE.Dec}(\mathsf{esk}_i, [\![\mathsf{x}_2]\!])$.
3. Compute $\mathsf{y}_1 = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{u}} \rangle$, $\mathsf{y}_2 = \mathsf{AHE.Dec}(\mathsf{esk}_i, [\![\mathsf{y}_2]\!])$.
4. Compute $\mathsf{z}_1 = \mathsf{AHE.Dec}(\mathsf{esk}_i, [\![\mathsf{z}_1]\!])$, $\mathsf{z}_2 = \mathsf{AHE.Dec}(\mathsf{esk}_i, [\![\mathsf{z}_2]\!])$.
5. Generate and send $\mathsf{msg}_3 = \{\mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{rec}} \leftarrow \mathsf{OT.Round}_1(\mathsf{crs}_i, \mathsf{s}_\mathsf{t})\}_{\mathsf{s} \in \{\mathsf{x},\mathsf{y},\mathsf{z}\}, \mathsf{t} \in \{1,2\}}$.

Round 4: $(\mathcal{P}_j \rightarrow \mathcal{P}_i)$ Party $\mathcal{P}_j$ does the following:

1. Compute $\widetilde{\mathcal{C}} = \mathsf{Garble}(\mathcal{C})$ for the circuit $\mathcal{C}$ described in Figure 6.
2. For each $\mathsf{s} \in \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}, \mathsf{t} \in \{0,1\}$, let $\mathsf{lab}_{\mathsf{s},\mathsf{t}}^0, \mathsf{lab}_{\mathsf{s},\mathsf{t}}^1$ denote the labels of the garbled circuit $\widetilde{\mathcal{C}}$ corresponding to input wires s_t. Generate $\mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{sen}} = \mathsf{OT.Round}_2(\mathsf{crs}_i, \mathsf{lab}_{\mathsf{s},\mathsf{t}}^0, \mathsf{lab}_{\mathsf{s},\mathsf{t}}^1, \mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{rec}})$. Let $\mathsf{ot}^{\mathsf{sen}} = \{\mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{sen}}\}_{\mathsf{s} \in \{\mathsf{x},\mathsf{y},\mathsf{z}\}, \mathsf{t} \in \{1,2\}}$
3. Compute $\mathsf{pad} = \mathsf{PRF}(\mathsf{k}_i, \mathsf{msg}_3)$. Set $\mathsf{ct}_j = \mathsf{SKE.Enc}(\mathsf{pad}, \mathsf{TS.Sign}(\mathsf{sk}_j^{\mathsf{TS}}, \mathsf{msg}))$.
4. Send $(\widetilde{\mathcal{C}}, \mathsf{ot}^{\mathsf{sen}}, \mathsf{ct}_j)$ to $\mathcal{P}_i$.

Round 4: $(\mathcal{P}_k \rightarrow \mathcal{P}_i)$ Party $\mathcal{P}_k$ does the following:

1. Compute $(\widetilde{\mathcal{C}}, \mathsf{ot}^{\mathsf{sen}}, \mathsf{pad})$ exactly as done by $\mathcal{P}_j$.
2. Set $\mathsf{ct}_k = \mathsf{SKE.Enc}(\mathsf{pad}, \mathsf{TS.Sign}(\mathsf{sk}_k^{\mathsf{TS}}, \mathsf{msg}))$.
3. Send $(\mathsf{RO}(\widetilde{\mathcal{C}}, \mathsf{ot}^{\mathsf{sen}}), \mathsf{ct}_k)$ to $\mathcal{P}_i$.

Output Computation: Parties $\mathcal{P}_j, \mathcal{P}_k$ output $(\mathsf{msg}, \mathcal{P}_i, S)$. Party $\mathcal{P}_i$ does:

1. Let $(\widetilde{\mathcal{C}}, \mathsf{ot}^{\mathsf{sen}}, \mathsf{ct}_j)$ be the message received from $\mathcal{P}_j$ and $(\mathsf{msg}_4, \mathsf{ct}_k)$ be the message received from $\mathcal{P}_k$. Abort if $\mathsf{RO}(\widetilde{\mathcal{C}}, \mathsf{ot}^{\mathsf{sen}}) \neq \mathsf{msg}_4$.
2. For each $\mathsf{s} \in \{\mathsf{x}, \mathsf{y}, \mathsf{z}\}, \mathsf{t} \in \{0,1\}$, compute $\mathsf{lab}_{\mathsf{s},\mathsf{t}} = \mathsf{OT.Output}(\mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{sen}}, \mathsf{ot}_{\mathsf{s},\mathsf{t}}^{\mathsf{rec}}, \mathsf{r}_{\mathsf{s},\mathsf{t}}^{\mathsf{ot}})$. Let $\mathsf{lab} = \{\mathsf{lab}_{\mathsf{s},\mathsf{t}}\}_{\mathsf{s} \in \{\mathsf{x},\mathsf{y},\mathsf{z}\}, \mathsf{t} \in \{0,1\}}$. Compute $\mathsf{pad} = \mathsf{Eval}(\widetilde{\mathcal{C}}, \mathsf{lab})$.
3. Compute $\mathsf{Token}_j = \mathsf{SKE.Dec}(\mathsf{pad}, \mathsf{ct}_j)$, $\mathsf{Token}_k = \mathsf{SKE.Dec}(\mathsf{pad}, \mathsf{ct}_k)$, $\mathsf{Token}_i = \mathsf{TS.Sign}(\mathsf{sk}_i^{\mathsf{TS}}, \mathsf{msg})$, $\mathsf{Token} \leftarrow \mathsf{TS.Combine}(\{\mathsf{Token}_s\}_{s \in \{i,j,k\}})$.

Inputs: $(x_1, x_2, y_1, y_2, z_1, z_2)$. **Hardwired values:** $(a, b, e, f, p, q, r_z, \mathsf{pad}, d^2)$.
Computation:

- Abort if $x_2 \neq (a \cdot x_1 + b)$ (or) $y_2 \neq (e \cdot y_1 + f)$ (or) $z_2 \neq (p \cdot z_1 + q)$
- Compute $\mathsf{IP} = (z_1 - r_z) + x_1$
- If $\mathsf{IP}^2 \geq (d^2 \cdot y_1)$, output pad. Else, output $\perp$.

Fig. 6: Circuit $\mathcal{C}$ to be garbled.

4. Output $\{\mathsf{Token}_s\}_{s\in\{i,j,k\}}$ if $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token})$. Else, output $\perp$.

Token Verification: Given a verification key $\mathsf{vk}^{\mathsf{TS}}$, message msg and token $(\mathsf{Token}_i, \mathsf{Token}_j, \mathsf{Token}_k)$, the token verification algorithm does the following:

1. Compute $\mathsf{Token} \leftarrow \mathsf{TS.Combine}(\{\mathsf{Token}_s\}_{s\in\{i,j,k\}})$.
2. Output 1 if $\mathsf{TS.Verify}(\mathsf{vk}^{\mathsf{TS}}, \mathsf{msg}, \mathsf{Token}) = 1$. Else, output 0.

References

1. About Face ID advanced technology. https://support.apple.com/en-us/HT208108, accessed on March 2, 2021 2
2. Advantages and disadvantages of biometrics. https://www.ukessays.com/dissertation/examples/information-systems/advantages-and-disadvantages-of-biometrics.php?vref=1., accessed on March 2, 2021 2
3. FIDO Alliance. https://fidoalliance.org/, accessed on March 2, 2021 2
4. Google Pixel Fingerprint. https://support.google.com/pixelphone/answer/6285273?hl=en, accessed on March 2, 2021 2
5. iOS Security — iOS 12. https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf, page-8, Accessed on March 2, 2021 2
6. List of data breaches. https://en.wikipedia.org/wiki/List_of_data_breaches, accessed on March 2, 2021 2
7. NISTIR Draft on Ongoing Face Recognition Vendor Test Part 1: Verification. https://pages.nist.gov/frvt/reports/11/frvt_report_2020_01_21.pdf, accessed on March 2, 2021 6
8. Privacy Rights Clearinghouse – Data Breaches. https://www.privacyrights.org/data-breaches, accessed on March 2, 2021 2
9. Samsung Galaxy: Iris Scans for Security. https://www.samsung.com/global/galaxy/galaxy-s8/security/, accessed on March 2, 2021 2
10. Web Authentication: W3 Standard. https://www.w3.org/TR/2018/CR-webauthn-20180320/, accessed on March 2, 2021 2
11. White-Box Competition. https://whibox-contest.github.io/, accessed on March 2, 2021 2
12. Agrawal, S., Badrinarayanan, S., Mohassel, P., Mukherjee, P., Patranabis, S.: BETA: biometric enabled threshold authentication. IACR Cryptol. ePrint Arch. 2020, 679 (2020) 9, 12, 17, 18, 20, 23

13. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO (2001) 3
14. Beimel, A., Ishai, Y., Kushilevitz, E.: Ad hoc PSM protocols: Secure computation without coordination. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 580–608. Springer, Heidelberg (Apr / May 2017) 6
15. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Heidelberg (Apr / May 2018) 5, 7, 18
16. Blazy, O., Chevalier, C., Vergnaud, D.: Mitigating server breaches in password-based authentication: Secure and efficient solutions. In: CT-RSA (2016) 7
17. Blundo, C., De Cristofaro, E., Gasti, P.: Espresso: Efficient privacy-preserving evaluation of sample set similarity. In: Di Pietro, R., Herranz, J., Damiani, E., State, R. (eds.) Data Privacy Management and Autonomous Spontaneous Security (2013) 7
18. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (Jan 2003) 13, 18, 23
19. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1175–1191. ACM Press (Oct / Nov 2017) 7
20. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: CRYPTO (2018) 18, 20, 23
21. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004. pp. 82–91. ACM Press (Oct 2004) 3
22. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (Apr 2015) 6
23. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: Karlin, A.R. (ed.) ITCS 2018. vol. 94, pp. 21:1–21:21. LIPIcs (Jan 2018) 6
24. Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 190–213. Springer, Heidelberg (Aug 2016) 7
25. Bringer, J., Chabanne, H., Patey, A.: SHADE: Secure HAmming DistancE computation from oblivious transfer. In: Adams, A.A., Brenner, M., Smith, M. (eds.) FC 2013 Workshops. pp. 164–176. LNCS, Springer, Heidelberg (Apr 2013) 7
26. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001) 4, 15
27. Canetti, R.: Universally Composable Signature, Certification, and Authentication. In: CSFW (2004) 15
28. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: CRYPTO (2015) 15
29. Chan, T.H.H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 200–214. Springer, Heidelberg (Feb / Mar 2012) 7
30. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (May 2001) 5, 12, 23

31. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (Feb 2001) 5, 12, 23
32. Dinh, T., Steinfeld, R., Bhattacharjee, N.: A lattice-based approach to privacy-preserving biometric authentication without relying on trusted third parties. In: ISPEC (2017) 7
33. Dodis, Y., Reyzin, L., Smith, A.D.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: EUROCRYPT (2004) 3
34. Dupont, P.A., Hesse, J., Pointcheval, D., Reyzin, L., Yakoubov, S.: Fuzzy password-authenticated key exchange. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 393–424. Springer, Heidelberg (Apr / May 2018) 3, 6
35. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (Aug 1984) 12
36. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: 26th ACM STOC. pp. 554–563. ACM Press (May 1994) 6
37. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013) 3
38. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Heidelberg (Apr / May 2018) 5, 7, 18
39. Grassi, P.A., Fenton, J., Newton, E., Perlner, R., Regenscheid, A., Burr, W., Richer, J., Lefkovitz, N., Danker, J., Choong, Y., Greene, K., Theofanos, M.: Nist special publication 800-63b: Digital identity guidelines: Authentication and lifecycle management (June 2017), https://pages.nist.gov/800-63-3/sp800-63b.html 2
40. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. Journal of Cryptology 25(1), 158–193 (Jan 2012) 23
41. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: ISTCS '97. Washington, DC, USA (1997) 6
42. Joye, M., Libert, B.: A scalable scheme for privacy-preserving aggregation of time-series data. In: Sadeghi, A.R. (ed.) FC 2013. LNCS, vol. 7859, pp. 111–125. Springer, Heidelberg (Apr 2013) 7
43. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: CVPR (2017) 5
44. Mohassel, P., Rosulek, M., Zhang, Y.: Fast and secure three-party computation: The garbled circuit approach. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 591–602. ACM Press (Oct 2015) 6, 11, 13
45. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: EUROCRYPT (2016) 5, 7
46. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (May 2016) 18
47. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA. pp. 448–457. ACM-SIAM (Jan 2001) 23
48. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999) 12, 23
49. Peikert, C., Shiehian, S.: Multi-key FHE from lwe, revisited. In: TCC (2016) 5

50. Peikert, C., Shiehian, S.: Multi-key FHE from LWE, revisited. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 217–238. Springer, Heidelberg (Oct / Nov 2016) 7, 18
51. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: CRYPTO (2019) 20
52. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (Aug 2008) 23
53. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT (2005) 6
54. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR (2015) 5
55. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (May 2000) 18, 23
56. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: CVPR (2018) 5
57. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986) 11