Turning HATE into LOVE: Compact Homomorphic Ad Hoc Threshold Encryption for Scalable MPC

Leonid Reyzin[1], Adam Smith[1], and Sophia Yakoubov[2](✉)

[1] Boston University, Boston, USA
{reyzin,ads22}@bu.edu
[2] Aarhus University, Åbogade 34, 8200 Aarhus, Denmark
sophia.yakoubov@cs.au.dk

Abstract. In a public-key threshold encryption scheme, the sender produces a single ciphertext, and any $t+1$ out of n intended recipients can combine their partial decryptions to obtain the plaintext. *Ad hoc* threshold encryption (ATE) schemes require no correlated setup, enabling each party to simply generate its own key pair. In this paper, we initiate a systematic study of the possibilities and limitations of ad-hoc threshold encryption, and introduce a key application to scalable multiparty computation (MPC).

Assuming indistinguishability obfuscation (iO), we construct the first ATE that is *sender-compact*—that is, with ciphertext length independent of n. This allows for succinct communication once public keys have been shared. We also show a basic lower bound on the extent of key sharing: every sender-compact scheme requires that recipients of a message know the public keys of other recipients in order to decrypt.

We then demonstrate that threshold encryption that is ad hoc and *homomorphic* can be used to build efficient large-scale fault-tolerant multiparty computation (MPC) on a minimal (star) communication graph. We explore several homomorphic schemes, in particular obtaining one iO-based ATE scheme that is both sender-compact and homomorphic: each recipient can derive what they need for evaluation from a single short ciphertext. In the resulting MPC protocol, once the public keys have been distributed, all parties in the graph except for the central server send and receive only short messages, whose size is independent of the number of participants.

Taken together, our results chart new possibilities for threshold encryption and raise intriguing open questions.

Keywords: Threshold encryption · Obfuscation · Setup freeness · Secure computation

L. Reyzin—Supported in part by NSF grant 1422965.
A. Smith—Supported in part by NSF awards IIS-1447700 and AF-1763786 and a Sloan Foundation Research Award.
S. Yakoubov—Work done while at Boston University, and was supported in part by NSF grant 1422965.

1 Introduction

A public key threshold encryption (TE) scheme gives one the ability to generate a ciphertext that is decryptable by any $t+1$ out of n intended recipients, while remaining semantically secure against any smaller group. Among other things, it enables tasks such as electronic voting [16,17] and round-efficient multiparty computation (MPC) [2,19], where only $t+1$ colluding parties should be able to learn information about others' inputs.

One simple way to construct threshold encryption is to use any encryption scheme, with each of n recipients having independently generated keys. To encrypt, the sender applies $(t+1, n)$-secret sharing to the message, and encrypts each share with the key of the respective recipient; we call this *share-and-encrypt*.

Share-and-encrypt has the advantage of requiring no master secret and no correlated setup among the recipients. A basic public-key infrastructure is all that is required. We will call TE schemes with this property *ad hoc* threshold encryption (ATE). An additional advantage of this simple approach is that the length of information sent to each recipient is independent of the number of recipients (since each recipient needs to see only the part of the ciphertext relevant to them). We will call TE with this property *recipient-compact*. It is, however, not *sender-compact*, because the length of information sent by the sender is dependent on the number of recipients. This missing feature is particularly desirable when the sender, rather than unicasting information to each recipient, broadcasts it—for example, by using an intermediate server. Prior to this paper, whether sender-compactness is achievable for ad hoc TE was an open problem.

1.1 Our Contributions

In this paper, we initiate a systematic study of the possibilities and limitations of ad hoc threshold encryption, and introduce a key application to scalable MPC. We start with a definitional framework that systematizes the various options for functionality and security in Sect. 2.

As our main feasibility result (Sect. 3), we show that sender-compactness is, in principle, achievable.

Contribution 1 (Theorem 1). *We describe the first sender-compact ad hoc threshold encryption scheme.*

The price we pay for sender-compactness is that we use indistinguishability obfuscation (iO), and that every sender needs a public key. This key needs to be known for decryption, and has a component whose size grows polynomially with n. However, public keys are published once, whereas ciphertexts are created and transmitted multiple times, so having the burden of size in the public keys instead of the ciphertexts can be a big advantage when the sender is already known to the recipient. Moreover, in some uses of TE, decryption is delayed, and the linear component of the public key is not needed by every recipient (for example, if TE is used for backup storage that is usually not accessed; see Sect. 1.2 for another example).

We also show a fundamental limitation of sender-compact schemes: recipients need to know the public keys of other recipients. Specifically, we define (in Sect. 2) a TE property we call recipient-set-obliviousness, which demands that the recipient algorithms be run without the public keys of other recipients.

Contribution 2 *We show that recipient-set-obliviousness and sender-compactness cannot be simultaneously satisfied.*

We formally state and prove this result in the full version of this paper.

Threshold encryption is well suited for applications to multi-party computation (MPC), because it allows multiple parties to learn shares of a value. Building MPC protocols is much easier when encryption also allows for some homomorphic computation, so that operations on unopened ciphertexts can be used to operate on the underlying plaintexts.

We demonstrate, in Sect. 4, how to build recipient-compact ad hoc threshold encryption schemes that support limited homomorphism. We use the acronym "HATE" to describe ATE schemes that support homomorphism.

Contribution 3 (Theorems 3, 4 and 5). *We describe three recipient-compact HATE schemes that support additive homomorphism.*

The first two of these schemes are based on standard assumptions. They follow the share-and-encrypt paradigm, and allow homomorphism because of a careful combination of specific encryption and secret sharing schemes. One of these schemes keeps messages in the exponent, and thus supports only limited message spaces. Choosing a secret sharing scheme with the right properties is crucial to enable the scheme to be ad hoc, recipient-compact, and homomorphic. We use Shamir and CRT secret sharing, both of which are additively homomorphic over multiple inputs, do not require pre-distributed correlated randomness, and have compact shares.

These schemes are recipient-set-oblivious and therefore cannot be sender-compact. However, they have an additional property on top of recipient compactness, which we call *recipient-local evaluation*: namely, not only does a ciphertext consists of compact recipient-wise components, but also each recipient can perform homomorphic evaluation locally on its own components.

The third recipient-compact additively homomorphic ATE has the advantage that a fresh ciphertext (before homomorphic evaluation) is sender-compact, but at the price of relying on iO. We obtain this scheme by modifying our iO-based scheme from Sect. 3. Prior to homomorphic evaluation, a different ciphertext (of size independent of n) for each recipient must be extracted from the sender-compact ciphertext. As in the first two schemes, homomorphic evaluation can be performed locally on these per-recipient ciphertexts, giving the scheme *recipient-local evaluation.* This scheme supports only small message spaces.

Open Questions About Ad hoc Threshold Encryption. Our systematic study and results raise several intriguing open problems about ad hoc threshold encryption. First, are there sender-compact ad hoc threshold encryption schemes with constant-size public keys (independent of n)? Are there such schemes which do not require a sender public key? Can such schemes be based on more standard assumptions than iO? Are there ad hoc threshold encryption schemes with ciphertexts that remain compact even after homomorphic evaluation? Is it possible to achieve full homomorphism? (We note that share-and-encrypt is not known to solve this problem: in principle, a multi-input fully homomorphic threshold secret sharing scheme can be combined with fully homomorphic encryption to give fully homomorphic ad hoc threshold encryption; however, to the best of our knowledge, all known constructions of multi-input fully homomorphic threshold secret sharing require pre-distributed correlated randomness.)

The importance of our results and these questions is reinforced by their usefulness for scalable MPC, which we discuss next.

1.2 Application: One-Server, Fault-Tolerant MPC

Consider a service that has an app with a large smartphone user base. Suppose the service wants to collect aggregate usage statistics, but (for regulatory compliance, or for good publicity, or for fear of becoming a target for attackers and investigators) does not wish to learn the data of any individual user.

A traditional MPC solution is not suitable for this setting, because the phones do not communicate directly with one another, and because we cannot expect every phone to remain engaged for the duration of the protocol, as phones may go out of signal range or run out of charge. We call MPC protocols in this setting Large-scale One-server Vanishing-participants Efficient MPC (LOVE MPC).

As we already mentioned, threshold encryption can be used for MPC. Ad hoc threshold encryption is particularly well-suited for this setting: by not having a setup phase, it eliminates an important bottleneck, because running a multi-user setup protocol with vanishing participants may present problems. In particular, HATE schemes can be used to build LOVE MPC for the honest-but-curious setting as follows: each phone sends an encryption of its input to the server, who homomorphically combines them, sends the result out for decryption by all users, and successfully uses the partial decryptions to get the correct result as long as more than t phones respond.

Using our HATE constructions, we derive a 3-round LOVE MPC for addition (described in the full version of this paper). This improves on the round complexity of prior work by Bonawitz *et al.* [5], who proposed a 5-round protocol. (We also prove, in the full version of this paper, that three rounds and some setup—e.g. a PKI—is necessary for LOVE MPC.)

The resulting LOVE MPC is based on standard assumptions when using the HATE constructions of Sect. 4.1, and the linear per-user communication we obtain asymptotically matches the per-user communication of Bonawitz *et al.* [5]. (Per-user communication was improved to constant by subsequent work of Bell *et al.* [4], but still at the cost of 5 rounds as opposed to our 3.) Additionally, at the price of using our iO-based HATE construction (Section Sect. 4.2), we obtain constant per-user communication, which is asymptotically better than the protocol of Bonawitz *et al.* [5] and asymptotically matches the protocol of Bell *et al.* [4] (but, of course, at very high concrete costs due to the use of iO).

1.3 Related Work

Threshold Encryption. Known sender-compact threshold encryption schemes are not ad hoc: they require some correlated setup. For instance, a sender-compact threshold variant of ElGamal, due to Desmedt and Frankel [13] (and described in the full version of this paper) requires a setup phase for every new set of n recipients. Delerablée and Pointcheval [12] designed a sender-compact scheme based on bilinear maps with a reduced setup requirement. In their scheme, the sender can pick the set of n recipients dynamically; however, each recipient's secret key must be derived from a common master secret key, so this scheme is not ad hoc.

On the other hand, known ad hoc threshold encryption schemes are not sender-compact. The simple share-and-encrypt construction discussed above requires the sender to send an amount of information that is linear in n. Daza *et al.* [11] use an interpolation-based trick to reduce the ciphertext size to $O(n-t)$ (and subsequently use bilinear maps to give a matching CCA2-secure construction [10]); however, they leave open the problem of further lowering the ciphertext size.

Ad hoc *fully homomorphic* threshold encryption was explored by Boneh *et al.* [6] and Badrinarayanan *et al.* [2], as well as by Dodis *et al.* [14] as "spooky" encryption; however, their schemes are not even recipient-compact, let alone sender-compact.

Ad Hoc Broadcast Encryption. Ad hoc sender-compact encryption has been achieved in the context of broadcast encryption, which is a special case of threshold encryption with the threshold $t = 0$, giving any one recipient the ability to decrypt. Specifically, Boneh and Zhandry [7] construct what they call *distributed* broadcast encryption form indistinguishability obfuscation (iO). Their construction has the downside of long (polynomial in the number n of recipients) public keys. Later, Ananth et al. [1] shrink the public keys at the cost of changing the assumption to differing-inputs obfuscation (diO). Zhandry [21] improves on these results, shrinking the public keys and replacing the iO assumption with witness PRFs, but still requiring $t = 0$.

2 Threshold Encryption (TE) Definitions

A threshold encryption scheme [13] is an encryption scheme where a message is encrypted to a group $\mathcal{R}$ of recipients, and decryption must be done collaboratively by at least $t + 1$ members of that group. (This can be defined more broadly for general access structures, but we limit ourselves to the threshold access structure in this paper.)

Classically, threshold cryptography involves a secret-shared secret key, which fixes the set of all key-holders. That is, a single Setup operation suffices only to establish a single set of recipients, and the sender is not allowed to specify a recipient set $\mathcal{R}$ at encryption time.

Dynamic threshold encryption [12] allows a sender to choose the set of recipients dynamically at encryption time, as described in the Enc algorithm of Sect. 2.1. In a dynamic threshold encryption scheme, a single Setup operation suffices for the establishment of arbitrarily many groups of recipients.

However, dynamic threshold encryption schemes still require trusted setup, where a central authority distributes correlated randomness to all parties. In an *ad hoc* threshold encryption (ATE) scheme, there is no need for any trusted central authority or master secret key msk. We call a threshold encryption scheme ad hoc if a public-private key pair can be generated without knowledge of a master secret key; that is, if each party is able to generate its keys independently.

In this paper, we additionally consider *keyed-sender* threshold encryption schemes. In a keyed-sender threshold encryption scheme, in order to encrypt a message, the sender must use its own *secret key* in addition to the recipients' public keys (unlike in typical public-key encryption, where encryption does not require the knowledge of any secrets). Similarly, in order to decrypt the ciphertext, recipients need to use the sender's public key in addition to their secret keys.

2.1 Threshold Encryption Syntax

A threshold encryption scheme consists of five algorithms, described in this section. This description is loosely based on the work of Daza *et al.* [10], but we modify the input and output parameters to focus on those we require in our constructions, with some additional parameters discussed in the text. Parameters in purple (namely, msk) are absent from ad hoc schemes; parameters in blue (namely, sk_{Sndr} and pk_{Sndr}) are present only in keyed-sender schemes (for readers seeing this text in monochrome, we give text explanations in addition to colors). Keyed-sender schemes additionally require a sixth algorithm, $\mathsf{KeyGen}_{\mathsf{Sndr}}$.

$\mathsf{Setup}(1^\lambda, t) \rightarrow (\mathsf{params}, \mathsf{msk})$ is a randomized algorithm that takes in a security parameter λ as well as a threshold t and sets up the global public parameters params for the system.

If the scheme is not ad hoc, Setup also sets up the master secret key msk for key generation.

For simplicity, we provide Setup with the threshold t, and assume that t is encoded in params. However, in *t-flexible* schemes, t may be decided by each sender at encryption time, and should then be an input to Enc (and encoded in the resulting ciphertext). In keyed-sender schemes (where the sender must use their secret key to encrypt and recipients must use the sender's public key to decrypt), t may also be specified in the sender's public key.

$\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}) \rightarrow (pk, sk)$ is a randomized key generation algorithm that takes in the global public parameters params (and, if the scheme is not ad hoc, the master secret key msk) and returns a recipient's public-private key pair.

$\mathsf{KeyGen}_{\mathsf{Sndr}}(\mathsf{params}, \mathsf{msk}) \rightarrow (pk_{\mathsf{Sndr}}, sk_{\mathsf{Sndr}})$ is a randomized algorithm present in keyed-sender schemes only; it takes in the global public parameters params (and, if the scheme is not ad hoc, the master secret key msk) and returns a sender's public-private key pair where the private key is used to facilitate encryption by the sender, the public key is used to facilitate decryption of messages from the sender.

$\mathsf{Enc}(\mathsf{params}, sk_{\mathsf{Sndr}}, \{pk_i\}_{i \in \mathcal{R}, |\mathcal{R}|>t}, m) \rightarrow c$ is a randomized encryption algorithm that encrypts a message m to a set of public keys belonging to the parties in the intended recipient set $\mathcal{R}$ in such a way that any size-$(t+1)$ subset of the recipient set should jointly be able to decrypt. We assume t is specified within params, but (if the scheme is keyed-sender) it may also be specified within the sender's public key, or (if the scheme is t-flexible) it may be specified on the fly as an input to Enc itself.

$\mathsf{PartDec}(\mathsf{params}, pk_{\mathsf{Sndr}}, \{pk_i\}_{i \in \mathcal{R}}, sk_j, c) \rightarrow d_j$ is an algorithm that uses a secret key sk_j belonging to one of the intended recipients (that is, for $j \in \mathcal{R}$) to get a partial decryption d_j of the ciphertext c. This partial decryption can then be combined with t other partial decryptions to recover the message.

$\mathsf{FinalDec}(\mathsf{params}, pk_{\mathsf{Sndr}}, \{pk_i\}_{i \in \mathcal{R}}, c, \{d_i\}_{i \in \mathcal{R}' \subseteq \mathcal{R}, |\mathcal{R}'|>t}) \rightarrow m$ is an algorithm that combines $t+1$ or more partial decryptions to recover the message m.

In a sender-compact scheme, the size of the ciphertext c is independent of the number of recipients n. In a recipient-compact scheme, $\mathsf{PartDec}$ requires only a portion c_i of the ciphertext c, where the size of c_i is independent of n.

2.2 Threshold Encryption Flexibility

Not all threshold encryption schemes allow/require all of the algorithm inputs described in Sect. 2.1. Sometimes disallowing an input can make the scheme less flexible, but, on the other hand, sometimes schemes that do not rely on certain inputs have an advantage.

More Flexibility: Unneeded Inputs. Ad hocness is an example of gaining an advantage by eliminating dependence on an input. Ad hoc schemes do not use the master secret key msk, and thus do not require a trusted central authority (which in many scenarios might not exist).

Another example of gaining an advantage by eliminating an input is *recipient-set-obliviousness.* Requiring both decryption algorithms ($\mathsf{PartDec}$ and $\mathsf{FinalDec}$) to be aware of the set of public keys belonging to individuals in the set $\mathcal{R}$ of recipients can be limiting.

Definition 1 (Threshold Encryption: Recipient-Set-Obliviousness). *We call a threshold encryption scheme* recipient-set-oblivious *if neither partial decryption nor final decryption use* $\{pk_i\}_{i \in \mathcal{R}}$.

It may seem that a recipient-set-oblivious scheme should require less communication, since the sender would never need to communicate $\mathcal{R}$ to the recipients. However, in the full version of this paper we show that a recipient-set-oblivious ATE scheme cannot be sender-compact.

More Flexibility: Additional Inputs. In describing the threshold encryption algorithms, for the most part we assumed that the threshold t was fixed within the global public parameters params (or, in a keyed-sender scheme, in the sender's public key). However, some schemes (such as share-and-encrypt) allow the sender to choose t at encryption time; we call such schemes *t-flexible*.

2.3 Threshold Encryption Security

The threshold encryption security definition is two-fold. We require *semantic security*, informally meaning that encryptions of two messages of the same size should be indistinguishable. We use the semantic security definition of Boneh *et al.* [6] for threshold encryption schemes, modified to support the keyed-sender property. We also require *simulatability*, informally meaning that given a ciphertext corresponding to one of two messages, partial decryptions can be simulated in such a way as to cause the ciphertext to decrypt to either of the two messages. The latter requirement is useful for MPC applications.

Both for semantic security and simulatability, there are three notions of security we consider, which differ according to the point in the security game at which the adversary must commit to the set of corrupt parties $\mathcal{C}$, and the set of challenge ciphertext recipients $\mathcal{R}$. From weakest to strongest, these are *super-static*, *static* and *adaptive* security. In *super-static* security, which is what our obfuscation-based construction achieves, the adversary specifies both $\mathcal{C}$ and $\mathcal{R}$ before seeing the public keys. In *static* security, which is what our other constructions achieve, the adversary specifies $\mathcal{C}$ before seeing the public keys, but can specify $\mathcal{R}$ later, at the same time as the two challenge messages, m_R and m_L. In *adaptive* security, the adversary specifies $\mathcal{C}$ having seen the public keys, and can specify $\mathcal{R}$ at the same time as the two challenge messages, as in static security.

The formal definitions of threshold encryption security are straightofrward given the above discussion, but are too lengthy given the space constraints. We therefore give them in the full version of this paper.

2.4 Threshold Encryption with Homomorphism

Homomorphic ad hoc threshold encryption (HATE) can be particularly useful in applications to multi-party computation.

Definition 2 (Threshold Encryption: Homomorphism). *Let $\mathcal{F}$ be a class of functions, each taking a sequence of valid messages and returning a valid message. An $\mathcal{F}$-*homomorphic *threshold encryption scheme additionally has the following algorithm:*

$\mathsf{Eval}(\mathsf{params}, \{pk_i\}_{i \in \mathcal{R}}, [c_1, \dots, c_\ell], f) \rightarrow c^*$ *is an algorithm that, given* ℓ *ciphertexts and a function* $f \in \mathcal{F}$*, computes a new ciphertext* c^* *which decrypts to* $f(m_1, \dots, m_\ell)$ *where each* c_q*,* $q \in [1, \dots, \ell]$ *decrypts to* m_q*.*

Informally, Eval should be *correct*, meaning that decryption should lead to the correct plaintext message $f(m_1, \dots, m_\ell)$.

2.5 Threshold Encryption Compactness

Compactness Without Homomorphism. As described in the introduction, we say that a threshold encryption scheme is *sender-compact* (or, in other words, that it has *sender-compact encryption*) if the size of a ciphertext is independent of the number of recipients. We say that it is *recipient-compact* (or, in other words, that it has *recipient-compact encryption*) if the portion of the ciphertext required by each recipient to produce their partial decryption is independent of the number of recipients. Of course, if a threshold encryption scheme is sender-compact, then it is also recipient-compact, since each receiver can use the entire (compact) ciphertext to partially decrypt. However, the converse is not necessarily true. Even if a scheme is not sender-compact, it can be recipient-compact if the ciphertext c can be split into compact components $c = \{c_i\}_{i \in \mathcal{R}}$ such that every recipient can run PartDec given just one component c_i.

Compactness With Homomorphism. When we consider homomorphic threshold encryption, a fresh ciphertext c may look different than a ciphertext c^* which Eval outputs. Of course, the size of c^* should not grow linearly with the number ℓ of inputs to f; otherwise, homomorphism becomes unnecessary, and c^* could simply consist of a concatenation of the input ciphertexts.

Notice that this does not preclude ciphertext growth. Even if a fresh ciphertext has size independent of n, the output of Eval may grow with n. We introduce some new terminology to handle this: we say that a homomorphic threshold encryption scheme has *compact evaluation* if the output of Eval has size independent of n, and that it has *recipient-compact evaluation* if the output of Eval can be split into recipient-wise compact components. Additionally, we say that a homomorphic threshold encryption scheme has *recipient-local evaluation* if it has *recipient-compact encryption* and evaluation is performed component-wise, with Eval taking one recipient's component of each input ciphertext and producing that recipient's compact component of the output ciphertext.

All of our schemes in Sect. 4 have recipient-compact encryption and recipient-local evaluation; the scheme in Sect. 4.2 additionally has sender-compact encryption.

In a setting where multiple senders send ciphertexts to a single server, who homomorphically computes on the ciphertexts and sends (the relevant parts of) the output of Eval to receivers, it is enough to have a sender-compact encryption and recipient-compact evaluation, even if the overall output of Eval is long. These properties suffice for reducing bandwidth, because the size of every message transmitted between two parties is independent of the number of recipients.

If, instead, we have a setting where senders send ciphertexts directly to receivers who then compute on those ciphertexts themselves, sender-compact encryption is less important, and recipient-local evaluation becomes key. Each sender must send something to each receiver anyway (instead of sending only one thing to the server), and in a setting with direct peer-to-peer channels, it becomes unimportant whether those things are all the same sender-compact ciphertext, or receiver-wise components of a recipient-compact ciphertext.

3 Sender-Compact Ad Hoc Threshold Encryption

In this section, we describe a sender-compact ATE. In the share-and-encrypt construction, the total ciphertext size is $\Theta(n)$, because each recipient gets an encryption of a different share. A natural approach is to compress the ciphertext using obfuscation: namely, instead of using the encrypted shares as the ciphertext, we can try to use an obfuscated program that *outputs* one encrypted share at a time given an appropriate input (such as a short symmetric encryption of the message, a recipient secret key, and proof of the recipient membership in the recipient set $\mathcal{R}$).

However, this strategy fails to achieve sender-compact ciphertexts, because the obfuscated program remains linear in the size of the threshold t. The reason is that, within the security proof, in one of the hybrids we are forced to hardcode t secret shares in the program, and the obfuscated program must be of the same size in all hybrid games.

Therefore, instead of putting an obfuscated program in the ciphertext, each sender obfuscates a program as part of key generation. This program becomes the sender's public key. While it is long (polynomial in the in the number of recipients n), it needs to be created and disseminated only once, as opposed to a ciphertext, which depends on the message. Notice that having this obfuscated program as the sender's public key makes our ATE scheme *keyed-sender*, meaning that in order to encrypt a message the sender must use its secret key, and in order to decrypt a message, recipients must use the sender's public key.

One can think of the obfuscated program in the sender's public key as a "horcrux".[1] The sender stores some of its secrets in this obfuscated program, and when encrypting a message, the sender includes just enough information in the ciphertext that the obfuscated program can do the rest of the work.

Once we put the obfuscated program in the sender's public key, we run into the issue that the outputs of the program on the challenge ciphertext cannot be dependent on the challenge message. This is because in the proof of security, the challenge message is chosen dynamically by the adversary, whereas the program is obfuscated by the challenger at the beginning of the game. In some hybrids, the outputs corresponding to the challenge message must be hardcoded in the program; so, they cannot depend on the actual message, which can be picked after the program is fixed. Therefore, instead of returning secret shares of the

[1] A "horcrux" is a piece of one's soul stored in an external object, according to the fantasy series Harry Potter [20].

challenge message, the program returns shares of a random mask which is used to encrypt the message.

Specifically, the program that each sender obfuscates takes as input a random nonce—together with the sender's signature on that nonce—and a recipient's secret key. The program checks the signature, and that the recipient's secret key matches one of the public keys to which the sender addressed this ciphertext (this "addressing" is performed implicitly, via the same signature). Note that checking membership in the set of recipients is important: otherwise any party could extract a secret share of the message. If the checks pass, the program outputs a secret share of a PRF output on the random nonce. The actual message is symmetrically encrypted with that PRF output.

The obfuscated program that makes up the sender public key is formally described in Algorithm 1, and the obfuscation-based ATE is described in Construction 1. It uses an indistinguishability obfuscator iO, puncturable pseudorandom function PPRF, a secret sharing scheme SS, a constrained signature SIG, and a length-doubling pseudorandom generator PRG with domain $\{0,1\}^\lambda$ and range in $\{0,1\}^{2\lambda}$. We define all of these primitives in the full version of this paper.

Algorithm 1. $f_{k_w, k_{\mathsf{Share}}, \mathsf{SIG}.pk}(\overrightarrow{pv} = \{pv_i\}_{i \in \mathcal{R}}, \mathsf{idx}, sv, \mathsf{nonce}, \sigma)$

```
The following values are hardcoded:
  $\mathsf{params} = (\lambda, n, t)$, where
    $\lambda$ is the security parameter,
    $n$ is the number of recipients, and
    $t$ is the threshold.
  $k_w$, a secret PPRF key used to recover the mask $w$ from nonce $\mathsf{nonce}$
  $k_{\mathsf{Share}}$, a secret PPRF key used to secret share the mask $w$
  $\mathsf{SIG}.pk$, a signature verification key
The following values are expected as input:
  $\overrightarrow{pv} = \{pv_i \in \{0,1\}^{2\lambda}\}_{i \in \mathcal{R}}$, lexicographically ordered public values
  $\mathsf{idx}$, an index
  $sv \in \{0,1\}^\lambda$, a secret value
  $\mathsf{nonce}$
  $\sigma$, a signature
if ($\overrightarrow{pv}[\mathsf{idx}] = \mathsf{PRG}(sv)$) and ($\mathsf{SIG.Verify}(\mathsf{SIG}.pk, (\overrightarrow{pv}, \mathsf{nonce}), \sigma)$) then
  $w \leftarrow \mathsf{PPRF}_{k_w}(\mathsf{nonce})$
  $r \leftarrow \mathsf{PPRF}_{k_{\mathsf{Share}}}(\mathsf{nonce})$
  $[w]_{\mathsf{idx}} \leftarrow \mathsf{SS.Share}(w, n, t; r)[\mathsf{idx}]$ {This gives the idxth secret share of $w$}
  return $[w]_{\mathsf{idx}}$
```

Informally, in order to prove security, we will have to show that given an obfuscation of this program, an adversary who has only t or fewer secret keys from the recipient set will not be able to tell the difference between an encryption of a message m_R and an encryption of a different message m_L. Our proof will need to puncture k_w and k_{Share} on the challenge nonce in order to remove any

Let the public parameters $\mathsf{params} = (\lambda, n, t)$ consist of the security parameter λ, the number of recipients n, and the threshold t.

```
KeyGen(params):
    {The following generates the "receiver" keys.}
    $sv \leftarrow \{0,1\}^{\lambda}$
    $pv \leftarrow \mathsf{PRG}(sv) \in \{0,1\}^{2\lambda}$
    return $(pv, sv)$
$\mathsf{KeyGen}_{\mathsf{Sndr}}(\mathsf{params})$:
    {The following generates the "sender" keys.}
    $(\mathsf{SIG}.pk, \mathsf{SIG}.sk) \leftarrow \mathsf{SIG.KeyGen}(1^{\lambda})$
    $k_w \leftarrow \mathsf{PPRF.KeyGen}(1^{\lambda})$
    {This PPRF key will be used to produces the mask $w$ for the message. Its output is
    assumed to be in the message space group.}
    $k_{\mathsf{Share}} \leftarrow \mathsf{PPRF.KeyGen}(1^{\lambda})$
    {This PPRF key will be used to produce the randomness for secret sharing $w$. We slightly
    abuse PPRF notation above; the size of $w$ and the size of the randomness needed to
    secret share $w$ might be very different. We simply assume that either the keys used
    are of different sizes (that is, $k_{\mathsf{Share}}$ might actually consist of multiple keys), or that
    the PPRF is chained in the appropriate way to produce a sufficiently large amount of
    randomness. We assume that the output of PPRF with $k_w$ is in some group $\mathcal{G}$ which
    contains the message space, and that the output of PPRF with $k_{\mathsf{Share}}$ is of whatever form
    the randomness for SS.Share should take.}
    $\mathsf{ObfFunc} \leftarrow \mathsf{iO}(f_{k_w, k_{\mathsf{Share}}, \mathsf{SIG}.pk})$
    return $(pk_{\mathsf{Sndr}} = \mathsf{ObfFunc}, sk_{\mathsf{Sndr}} = (\mathsf{SIG}.sk, k_w))$
$\mathsf{Enc}(\mathsf{params}, sk_{\mathsf{Sndr}} = (\mathsf{SIG}.sk, k_w), \overrightarrow{pv} = \{pv_i\}_{i \in \mathcal{R}, |\mathcal{R}| \geq t}, m)$:
    $\mathsf{nonce} \leftarrow \mathsf{PPRF.domain}$
    $e = (\mathsf{PPRF}_{k_w}(\mathsf{nonce}) + m)$
    $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{SIG}.sk, (\overrightarrow{pv}, \mathsf{nonce}))$
    return $c = (\mathsf{nonce}, e, \sigma)$
$\mathsf{PartDec}(\mathsf{params}, pk_{\mathsf{Sndr}} = \mathsf{ObfFunc}, \overrightarrow{pv} = \{pv_i\}_{i \in \mathcal{R}}, sv_i, c = (\mathsf{nonce}, e, \sigma))$:

    Let idx be the index of the public value corresponding to the secret value $sv_i$ in a
    lexicographic ordering of $\{pv_i\}_{i \in \mathcal{R}}$
    $d_i \leftarrow \mathsf{ObfFunc}(\overrightarrow{pv}, \mathsf{idx}, sv_i, \mathsf{nonce}, \sigma)$
    return $d_i$
$\mathsf{FinalDec}(\mathsf{params}, c = (\mathsf{nonce}, e, \sigma), \{d_i\}_{i \in \mathcal{R}' \subset \mathcal{R}})$:
    $w \leftarrow \mathsf{SS.Reconstruct}(\{d_i\}_{i \in \mathcal{R}' \subset \mathcal{R}})$
    $m \leftarrow e - w$
    return $m$
```

Construction 1: Obfuscation-Based ATE

information about the challenge plaintext from the program. For the proof to go through given the guarantees of iO, it is crucial that, as we change the plaintext, the output does not change for any input—in particular, even if the adversary is able to forge a signature that ties the ciphertext to a wrong set of public keys. We ensure this property by using a constrained signature scheme SIG, so that we can guarantee (in an appropriate hybrid) that a signature tying the ciphertext to a wrong set of public keys does not exist. This means that the public verification key (which is incorporated into the obfuscated program) is of size polynomial in n.

Theorem 1. *The obfuscation-based ATE (Construction 1) is (n, t)-super-statically secure for any polynomial n, t, as long as* iO *is a secure indistinguishability obfuscator,* PPRF *is a secure puncturable* PRF, SS *is a secure (n, t)-secret* *sharing scheme,* SIG *is a constrained signature scheme, and* PRG *is a secure pseudorandom generator.*

We prove Theorem 1 in the full version of this paper. Note that all the tools this construction uses can be obtained from indistinguishability obfuscation (with complexity leveraging), and one-way functions.

3.1 *t*-Flexibility

For simplicity, we describe obfuscation-based ATE in a way that is not by default t-flexible, since the threshold t is fixed within the sender's public key. However, it can be made t-flexible in a very straightforward way, simply by including t as part of the (signed) input to the obfuscated program.

3.2 Reducing the Public Key Size

In the construction described above, the sender's public key size is polynomial in the number n of recipients. We can decrease the size of the public key by relying on differing-inputs obfuscation (diO) [1,3] instead of indistinguishability obfuscation (iO). If we do, then we can modify the obfuscated program to take a Merkle hash commitment to the set of recipients' public keys, instead of the entire list; additionally, we will be able to replace constrained signatures with any signature scheme. This will enable us to go from $poly(n)$-size public keys to $poly(t)$-size public keys. (We still need $poly(t)$ because that is the number of secret shares we must hard-code in the program in one of the hybrids in our security proof.)

4 Recipient-Compact Homomorphic Ad Hoc Threshold Encryption

In this section, we describe three recipient-compact HATE constructions. In addition to recipient-compactness, all three of these schemes have *recipient-local evaluation*, meaning that each recipient can perform evaluation locally given just their compact component of the ciphertext.

Two of them (Sect. 4.1) are based on the share-and-encrypt paradigm. These are recipient-set-oblivious, but are not sender-compact. The last (Sect. 4.2) achieves sender-compactness by combining share-and-encrypt with the obfuscation-based sender-compact ATE from Sect. 3. However, like the ATE in Sect. 3, it is not recipient-set-oblivious.

4.1 Building HATE from Homomorphic Encryption and Secret Sharing

In this section, we describe our share-and-encrypt homomorphic ad hoc threshold encryption scheme which, despite its $\Theta(n)$-size ciphertexts, is efficient enough to be used in practice in some scenarios, because it is recipient-compact.

As we mentioned in the introduction, one natural way to build ATE is to use a threshold secret sharing scheme SS together with a public-key encryption scheme PKE. The idea is to secret share the message, and to encrypt each share to a different recipient using their public key; therefore, we call this the *share-and-encrypt* paradigm. We elaborate on it in the full version of this paper.

Notice that we are able to omit all but the relevant part of the ciphertext as input to $\mathsf{PartDec}$ for each party (where the relevant part is the one encrypted under their key), making the scheme both recipient-set-oblivious and recipient-compact. This further saves on communication in some contexts.

Theorem 2. *Share-and-encrypt (described formally in the full version of this paper) is a (n, t)-statically secure, recipient-set-oblivious, recipient-compact ATE, as long as SS is a secure share simulatable t-out-of-n secret sharing scheme, and PKE is a CPA-secure public key encryption scheme.*

We prove Theorem 2 in the full version of this paper.

If the secret sharing and encryption schemes are homomorphic in compatible ways, the share-and-encrypt construction is a Homomorphic ATE. The trick is finding the right homomorphic secret sharing and encryption schemes. In particular, if the secret sharing scheme is $\mathcal{F}$-homomorphic, the encryption scheme must be $\mathcal{F}'$-homomorphic, where $\mathcal{F}'$ includes the homomorphic evaluation of $\mathcal{F}$ over secret shares.

Of course, if the secret sharing and encryption schemes are both *fully* homomorphic, they give fully homomorphic ATE. However, no homomorphic threshold secret sharing schemes (with homomorphism over multiple inputs, without pre-distributed correlated randomness) is known, to the best of our knowledge.[2]

We show two efficient combinations of secret sharing and encryption which result in additively homomorphic ATE: Shamir-and-ElGamal and CRT-and-Paillier(both described in detail in the full version of this paper).

Shamir-and-ElGamal. We build share-and-encrypt HATE out of ElGamal encryption [15] and a variant of Shamir secret sharing. We need to use a *variant* of Shamir secret sharing (which we call exponential Shamir secret sharing), and not Shamir secret sharing itself, because Shamir secret sharing is additively homomorphic (and the homomorphism is applied via addition of individual shares), so we would need the encryption scheme to support addition; however, ElGamal is only multiplicatively homomorphic, so if we attempt to apply a homomorphism on encrypted shares, it will not work. What we need in order to get an additively homomorphic ATE scheme is to use ElGamal encryption with a secret sharing scheme which is additively homomorphic, but whose homomorphism is applied via multiplication. Therefore, we need to alter our Shamir secret sharing scheme by moving the shares to the exponent; then, taking a product of two shares will result in a share of the sum of the two shared values. We refer to the full version of

[2] Boyle *et al.* [9] give a nice introduction to homomorphic secret sharing. Jain *et al.* [18] and Dodis *et al.* [14] both build (threshold) function secret sharing, which gives homomorphic secret sharing, but the homomorphism is only over a single input.

this paper for a description of the ElGamal encryption scheme and the exponential Shamir secret sharing scheme which we use.

Theorem 3. *Shamir-and-ElGamal (described in the full version of this paper) is an additively homomorphic ad hoc threshold encryption scheme for a polynomial-size message space.*

Shamir-and-ElGamal is an ad hoc threshold encryption scheme by Theorem 2; the homomorphism follows from the homomorphisms of the underlying encryption and secret sharing schemes.

In Shamir-and-ElGamal we are limited to polynomial-size message spaces since final decryption uses brute-force search to find a discrete log. Jumping ahead to LOVE MPC, polynomial-size message spaces are still useful in many applications, as explained in the introduction. Moreover, the server already does work that is polynomial in the number of users, so asking it to perform another polynomial computation is not unreasonable.

CRT-and-Paillier. We also build share-and-encrypt HATE out of Camenisch-Shoup encryption and Chinese Remainder Theorem based secret sharing. The Camenisch-Shoup encryption scheme is a variant of Paillier encryption that supports additive homomorphism. However, we cannot combine it with Shamir secret sharing, since Shamir shares all live in the same group, while each instance of a Camenisch-Shoup encryption scheme uses a different modulus. Therefore, we combine Camenisch-Shoup encryption with CRT secret sharing, which has exactly the property that different shares can live in different groups. Unlike Shamir-and-ElGamal, this HATE allows us to use large message spaces. We refer to the full version of this paper for a description of the Camenisch-Shoup encryption scheme and the CRT secret sharing scheme which we use.

Theorem 4. *CRT-and-Paillier (described in the full version of this paper) is an additively homomorphic ad hoc threshold encryption scheme.*

CRT-and-Paillier is an ad hoc threshold encryption scheme by Theorem 2; the homomorphism follows from the homomorphisms of the underlying encryption and secret sharing schemes.

4.2 Building HATE from Obfuscation

As described in Sect. 3, the obfuscation-based ATE is not homomorphic. Informally, in order to make the obfuscation-based ATE $\mathcal{F}$-homomorphic, we can modify the obfuscated program to:

1. Use a $\mathcal{F}$-homomorphic secret sharing scheme [8]. (As an example, Shamir secret sharing is additively-homomorphic.) Note that $\mathcal{F}$ should always include subtraction from a constant (in the appropriate group); the obfuscated program returns shares of the mask w, which we want to use, together with the masked message e, to obtain shares of $m = e - w$.

However, this alone is not enough; even if the secret shares returned by the obfuscated programs are homomorphic, in order to extract them from the ciphertext, one must know a recipient secret value sv, while evaluation should require no secrets.

2. Use $\mathcal{F}'$-homomorphic public key encryption and decryption keys pk_i, sk_i instead of public and private values $pv_i = \mathsf{PRG}(sv_i), sv_i$. The obfuscated program would then not require sk_i as input; instead, it would return a ciphertext that requires sk_i for decryption.
   $\mathcal{F}'$ must include the functions necessary to evaluate $\mathcal{F}$ on the homomorphic secret shares.

This modification makes the construction $\mathcal{F}$-homomorphic while preserving sender-compactness. Thus, anyone (e.g., a server) can evaluate the obfuscated program to extract encryptions of all recipients' shares of the mask, homomorphically convert these into encrypted shares of the message, and homomorphically compute on those encrypted shares (since our public key encryption scheme is homomorphic, as are secret shares). The server would then send all parties their encrypted share of the computation output. Additionally, this construction is recipient-compact (as long as homomorphic shares are small), since each party only needs one compact part of the ciphertext for partial decryption.

More concretely, we can use ElGamal encryption [15]. Once the obfuscated program is evaluated, we are essentially using the Shamir-and-ElGamal HATE (Sect. 4.1). In particular, this implies that we are limited to polynomial-size message spaces, since final decryption uses brute-force search to find a discrete logarithm.

In the full version of this paper we give more details about our homomorphic recipient-compact HATE construction.

Theorem 5. *The modified obfuscation-based ATE is (n, t)-super-statically secure for any polynomial n, t, as long as* iO *is a secure indistinguishability obfuscator,* PPRF *is a secure puncturable* PRF*,* SS *is a secure secret sharing scheme,* SIG *is a constrained signature scheme, and* PKE *is a secure public-key encryption scheme. Moreover, it is $\mathcal{F}$-homomorphic if* SS *is $\mathcal{F}$ homomorphic (where $\mathcal{F}$ includes subtraction from a constant), and if* PKE *is $\mathcal{F}'$-homomorphic (where $\mathcal{F}'$ includes the evaluation of $\mathcal{F}$ on* SS *secret shares).*

Acknowledgements. We would like to thank Ran Canetti and Ben Kreuter for helpful discussions.

References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptol. ePrint Arch. Rep. 2013, 689 (2013). http://eprint.iacr.org/2013/689
2. Badrinarayanan, S., Jain, A., Manohar, N., Sahai, A.: Secure MPC: laziness leads to GOD. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12493, pp. 120–150. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64840-4_5

3. Barak, B., et al.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
4. Bell, J., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly)logarithmic overhead. Cryptol. ePrint Archive Rep. 2020, 704 (2020). https://eprint.iacr.org/2020/704
5. Bonawitz, K., et al.: 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, New York (2017)
6. Boneh, D., et al.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 565–596. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_19
7. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_27
8. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 509–539. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_19
9. Boyle, E., Gilboa, N., Ishai, Y., Lin, H., Tessaro, S.: Foundations of homomorphic secret sharing. In: Karlin, A.R., (eds.), ITCS 2018, vol. 94, pp. 21:1–21:21. LIPIcs (Jan 2018)
10. Daza, V., Herranz, J., Morillo, P., Ràfols, C.: CCA2-secure threshold broadcast encryption with shorter ciphertexts. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 35–50. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75670-5_3
11. Daza, V., Herranz, J., Morillo, P., Ràfols, C.: Ad-hoc threshold broadcast encryption with shorter ciphertexts. Electr. Notes Theor. Comput. Sci. **192**(2), 3–15 (2008)
12. Delerablée, C., Pointcheval, D.: Dynamic threshold public-key encryption. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 317–334. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_18
13. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_28
14. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 93–122. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_4
15. 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 (1984)
16. Chaum, D. (ed.): Towards Trustworthy Elections. LNCS, vol. 6000. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12980-3
17. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_38
18. Jain, A., Rasmussen, P.M.R., Sahai, A.: Threshold fully homomorphic encryption. Cryptol. ePrint Arch. Rep. 2017, 257 (2017). http://eprint.iacr.org/2017/257

19. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_26
20. Rowling, J.: Harry Potter and the Half-Blood Prince. Bloomsbury (2005)
21. Zhandry, M.: How to avoid obfuscation using witness PRFs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 421–448. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_16