Public-Key Function-Private Hidden Vector Encryption (and More)

James Bartusek[1], Brent Carmer[2], Abhishek Jain[3], Zhengzhong Jin[3], Tancrède Lepoint[4], Fermi Ma[5], Tal Malkin[6], Alex J. Malozemoff[2], and Mariana Raykova[4]

[1] UC Berkeley[⋆⋆], `bartusek.james@gmail.com`
[2] Galois, `{bcarmer,amaloz}@galois.com`
[3] Johns Hopkins University, `abhishek@cs.jhu.edu,zjin12@jhu.edu`
[4] Google, `{tancrede,mariana}@google.com`
[5] Princeton University, `fermima@alum.mit.edu`
[6] Columbia University, `tal@cs.columbia.edu`

Abstract. We construct *public-key function-private* predicate encryption for the "small superset functionality," recently introduced by Beullens and Wee (PKC 2019). This functionality captures several important classes of predicates:

– Point functions. For point function predicates, our construction is equivalent to public-key function-private anonymous identity-based encryption.
– Conjunctions. If the predicate computes a conjunction, our construction is a public-key function-private hidden vector encryption scheme. This addresses an open problem posed by Boneh, Raghunathan, and Segev (ASIACRYPT 2013).
– d-CNFs and read-once conjunctions of d-disjunctions for constant-size d.

Our construction extends the group-based obfuscation schemes of Bishop et al. (CRYPTO 2018), Beullens and Wee (PKC 2019), and Bartusek et al. (EUROCRYPT 2019) to the setting of public-key function-private predicate encryption. We achieve an average-case notion of function privacy, which guarantees that a decryption key sk_f reveals nothing about f as long as f is drawn from a distribution with sufficient entropy. We formalize this security notion as a generalization of the (enhanced) real-or-random function privacy definition of Boneh, Raghunathan, and Segev (CRYPTO 2013). Our construction relies on bilinear groups, and we prove security in the generic bilinear group model.

1 Introduction

Predicate encryption [BW07, KSW08] is a powerful tool which enables fine-grained access to encrypted information. Roughly speaking, a sender can encrypt a message m (commonly referred to as a payload) with respect to an *attribute* x,

[⋆⋆] Research conducted at Princeton University.

while each decryption key sk_f is tied to a specific predicate f in some function class $\mathcal{F}$; the key sk_f correctly decrypts a ciphertext if and only if the associated attribute x satisfies $f(x) = 1$.

Generally, predicate encryption schemes simultaneously achieve *payload-hiding* and *attribute-hiding* security. At a high level, payload-hiding guarantees that an encryption of m with respect to attribute x reveals nothing about m to an adversary who does not possess a decryption key sk_f where $f(x) = 1$. Attribute-hiding guarantees that ciphertexts hide any information about x beyond what is leaked from successful decryption. That is, an adversary holding decryption keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_n}$ may learn the $0/1$ evaluations of $f_1, \ldots, f_n$ on x, but should not be able to learn anything else about x.

For certain applications, however, these two security guarantees may not be enough. Suppose an email user wants to set up a gateway that routes encrypted emails differently depending on whether or not they are spam. The user would like to avoid giving the gateway full access to the content of their emails; instead the user may have some list of potential spam email addresses, and would prefer the gateway only apply its spam filtering algorithm (which requires reading the email plaintext) if the email is sent from this set of addresses.[7]

The predicate encryption-based solution treats the contents of the email as the "payload," and the sender email address as the "attribute." The user generates some decryption key sk_f for their filtering predicate f (in this scenario, the predicate would output one if the email address belongs to a list of potential spammers), and sends this to the gateway. It is easy to imagine that the user may want to hide its particular choice of f from the gateway, since after all the user views the gateway as an untrusted party. But given sk_f, the standard payload-hiding and attribute-hiding definitions say nothing about whether one can learn the description of f.

Public-Key Function Privacy. Boneh, Raghunathan, and Segev (BRS) [BRS13a] address this problem by defining public-key *function-private* predicate encryption, which requires that sk_f leak nothing about f beyond what is leaked through honest decryption.[8] They demonstrate that this notion is achievable with a new construction of function-private anonymous identity-based encryption (i.e., predicate encryption for equality predicates). We note that here, "function private" means that the identity embedded in the decryption key is hidden, "anonymous" means that the intended recipient of the ciphertext (the attribute) is hidden; and finally the message being encrypted also stays secret. In follow-up work, BRS [BRS13b] extended public-key function privacy to a significantly larger class of *subspace-membership* predicates. We stress that in both works, BRS present function privacy as an average-case definition, which is essentially inherent in the public-key setting (see Section 1.2 for further discussion).

[7] Note that we would require a public-key predicate encryption scheme for this scenario, with the assumption that an email client would encrypt any email to the user under the user's public key.

[8] Function privacy had been studied before the work of BRS [BRS13a], albeit in the private key setting [SWP00, OS07, BSW09, SSW09].

While BRS [BRS13a, BRS13b] laid the groundwork for the study of function privacy in the public-key setting, a number of important questions remained unanswered. In particular, BRS explicitly identified three important directions for further exploration [BRS13b]:

1. **Computational Function Privacy.** In both works, BRS construct *statistically* function-private schemes. They conjectured, however, that it might be possible to leverage group-based assumptions to achieve more powerful/expressive *computationally* function-private predicate encryption schemes.
2. **Hidden Vector Encryption.** The seminal work of Boneh and Waters [BW07] introduced *hidden vector encryption* (HVE) as a general approach to performing equality, comparison, and subset queries on encrypted data. In HVE, predicates are specified by a vector $\mathbf{v} \in \Sigma^k$, where $\Sigma = \mathbb{Z}_s \cup \{*\}$. We refer to s as the alphabet size and $*$ as a *wildcard* character. A message m encrypted under attribute $\mathbf{x} \in \mathbb{Z}_s^k$ can be decrypted under key f_v if x_i matches v_i at each i where $v_i \neq *$. Follow-up work by Katz, Sahai, and Waters [KSW13] introduced *inner product encryption* (i.e., predicate encryption for inner product predicates) as a generalization of HVE. In turn, inner product predicates are a subclass of more general *subspace-membership predicates*. Therefore, predicate encryption for subspace-membership trivially implies inner product encryption and HVE.
   However, BRS [BRS13b] observe that these implications crucially do not preserve function privacy. That is, their function-private subspace-membership encryption construction is *not* a function-private HVE. In fact, BRS remark that even *defining* function privacy for HVE is not straightforward, and they leave defining and constructing function-private HVE as an open problem.
3. **Enhanced Function Privacy.** The plain definition of function privacy given by BRS [BRS13a] comes with a serious drawback. At a high level, the definition assumes that the adversary holding decryption key sk_f will never encounter a ciphertext with a matching attribute x (i.e., where $f(x) = 1$). The authors argue that such an assumption is necessary in many settings, since if an adversary could generate such matching ciphertexts, it must know some x where $f(x) = 1$. For equality predicates, this amounts to learning f entirely.
   In almost any natural application, however, we should expect that the party in possession of sk_f will encounter "matching" ciphertexts; the crucial point is that they would not be generating these ciphertexts themselves. To capture this, BRS define a stronger notion called *enhanced function privacy* where the adversary is given access to an "encryption oracle" that outputs matching ciphertexts.

   Unfortunately, the only known construction of a public-key scheme achieving enhanced function-privacy is the anonymous identity-based encryption construction presented by BRS [BRS13a]. Therefore, constructing enhanced-function-private predicate encryption schemes for any class of predicates beyond equality predicates has remained open since.

1.1 Our Contributions

In this work, we make substantial progress on all three fronts. Compared to BRS [BRS13a, BRS13b], our results come from using a qualitatively different high-level approach. In particular, BRS construct public-key function-private predicate encryption by starting from schemes that satisfy only data privacy (a definition combining attribute-hiding and payload-hiding), and transforming them to achieve data privacy and function privacy simultaneously.

We take the opposite approach. We begin with constructions that satisfy function privacy but not data privacy and transform them to achieve both data privacy and function privacy. In more standard terminology, our high-level approach is to think of an *obfuscated program* [BGI$^+$01] as a decryption key within a "predicate encryption" scheme that has no data privacy whatsoever (since obfuscated programs are run directly on non-encrypted inputs). We then show that several obfuscation schemes from the literature can be appropriately transformed to achieve public-key function-private predicate encryption.

Our starting point is the recent line of work [BKM$^+$18, BW19, BLMZ19] that constructs simple, group-based obfuscation schemes for what Beullens and Wee refer to as the "big subset" predicate [BW19].[9] For our work, we re-interpret these predicates as "small superset" predicates, a notion we find slightly more natural for our applications. A "small superset" predicate $f_{n,t,X}$ is parameterized by a target set $X \subseteq [n]$, an integer size bound $t \leq n$, and takes as input any set $Y \subseteq [n]$. $f_{n,t,X}(Y)$ outputs 1 if and only if $X \subseteq Y$ and $|Y| \leq t$ (that is, Y is a small superset of X). We show that "small superset" predicates capture several natural and expressive predicate classes, including large-alphabet conjunctions, functions in conjunctive normal form with a constant number of inputs per conjunct (a.k.a., d-CNFs for $d = O(1)$), and read-once conjunctions of d-disjunctions for $d = O(1)$.

Our primary contributions are the following:

1. We draw upon a correspondence between program obfuscation and function privacy to formulate new and versatile simulation-based definitions of average-case function-privacy and enhanced function-privacy.[10] While our definitions incorporate elements of the *distributional virtual black box* notion from obfuscation [BGI$^+$01, BR17], we view our (enhanced) function privacy definition as a natural extension of the definition of BRS [BRS13a]. Unlike these prior function-privacy notions [BRS13a, BRS13b], which are tailored to specific classes of predicates, our definition is completely agnostic to the predicate class.[11] For the special case of HVE (i.e., large-alphabet conjunc-

[9] We remark that [BKM$^+$18, BLMZ19] framed their results as obfuscation for conjunctions. Beullens and Wee [BW19] were the first to notice that these techniques are in fact obfuscating a more general class of "big subset" predicates, which in particular encompass conjunctions.

[10] While our definitions are new, we are not the first to observe the connection between program obfuscation and function-privacy. See also [AAB$^+$15, ITZ16, ABF16].

[11] We note that we are not the first to give a public-key function-private definition that is agnostic to the predicate class. In particular, this is also achieved by the definition

tions), we demonstrate that constructions achieving our function privacy definitions hide strictly more information about the underlying predicate than constructions achieving other recently proposed HVE function-privacy definitions (e.g., [PM18, PMR19]).

2. We leverage bilinear maps to construct a public-key predicate encryption scheme for small superset predicates. At a very high level, our construction works by embedding the group-based constructions developed in [BKM$^+$18, BW19, BLMZ19] in group $\mathbb{G}_1$, encoding messages/attributes in group $\mathbb{G}_2$, and decrypting using the bilinear map. We prove that our construction achieves enhanced function privacy in the generic bilinear group model. We note that generic analysis is somewhat unavoidable in our setting, as the underlying obfuscation constructions we build on are not known to be secure under any falsifiable assumption [Nao03, GW11].
3. We show that our general construction of public-key enhanced function-private predicate encryption for "small superset" immediately yields the following:
   - Anonymous IBE achieving enhanced function privacy as long as the underlying distribution on points has super-logarithmic min-entropy.
   - Public-key enhanced-function-private HVE whenever the underlying distribution meets a certain entropy threshold.
   - Public-key enhanced-function-private predicate encryption for d-CNFs and read-once conjunctions of d-disjunctions, subject to certain entropy requirements.

1.2 Technical Overview

Our Approach: From Obfuscation to Function-Private Predicate Encryption We begin by recalling the notion of *program obfuscation* [BGI$^+$01], which is the starting point for all of the constructions in this work. Roughly speaking, a program obfuscator takes in a description of some program P and outputs an obfuscated program $\mathsf{Obf}(P)$ that is functionally equivalent to P, but hides all of the implementation details. A natural approach to formalizing obfuscation security is the notion of a virtual black box (VBB), which asks that anything (precisely, any one-bit predicate) one can learn given $\mathsf{Obf}(P)$ can also be learned from black-box access to an oracle for P. While VBB obfuscation for general programs is known to be impossible [BGI$^+$01], there have been a number of positive results that achieve (average-case or worst-case) VBB security for limited classes of functionalities, such as point functions [Can97, LPS04, Wee05], conjunctions [BR13, BR17, BKM$^+$18, BLMZ19, BW19], Hamming balls [DS05], hyperplanes [CRV10], "compute-and-compare" functions [WZ17, GKW17], etc.

As mentioned in Section 1.1, there is a strong intuitive connection between program obfuscation and function-private predicate-encryption in the public-key

of [ITZ16]. However, their definition does not extend to enhanced function privacy, and furthermore they do not give any constructions achieving their definition except under a strengthening of indistinguishability obfuscation due to [BCKP14].

setting (this has also been observed in prior work [AAB$^+$15, ITZ16, ABF16]). In both settings, the goal is to allow evaluation of a specific functionality without leaking anything else about the functionality itself. The difference is that an obfuscated program runs on an arbitrary public input, while in function-private predicate encryption, function evaluation occurs when applying a decryption key for some predicate f to a ciphertext whose hidden attribute is the input to the function.

Moreover, we can imagine defining function privacy so that a decryption key sk_f for some function f leaks no more than a VBB obfuscation of f. In this case, public-key function-private predicate encryption for some function class $\mathcal{F}$ is a strictly stronger primitive than program obfuscation for $\mathcal{F}$. This follows trivially from the fact that anyone holding a decryption key sk_f for $f \in \mathcal{F}$ can use it as an obfuscated program: to learn whether $f(x)$ outputs 0 or 1, use the public key to encrypt a message payload under attribute x and check if decryption succeeds.

In this work, we leverage this intuitive connection to build public-key function-private predication encryption schemes by transforming simple obfuscators [Can97, BKM$^+$18, BLMZ19, BW19] that have appeared in the literature. Our core construction will be based on an obfuscator for the "small superset" functionality, which is essentially equivalent to the "big subset" functionality introduced by Beullens and Wee [BW19]. We define our "small superset" function $f_{n,t,X}$, parameterized by a target set $X \subseteq [n]$, a positive integer n, and an integer size bound $t \leq n$, to output 1 on input $Y \subseteq [n]$ if $X \subseteq Y$ and $|Y| \leq t$, and 0 otherwise.[12]

A simple group-based obfuscator for the "small superset" functionality follows easily from prior work [BLMZ19, BW19] (which are inspired by the construction of Bishop et al. [BKM$^+$18]). The obfuscation achieves an average-case notion of security (i.e., VBB holds if the set X is drawn from a distribution with appropriate entropy). From this, we build a public-key (enhanced) function-private predicate encryption scheme supporting the class of small superset predicates.

In the remainder of this technical overview, we describe a slightly simplified version of our construction to highlight the main ideas. Instead of starting with the "small superset" functionality, we use the simpler obfuscator of Canetti [Can97] for point functions. This yields a public-key function-private predicate encryption scheme for the equality predicate, or equivalently public-key anonymous IBE. We then provide an extensive discussion on our new definitions of function privacy and enhanced function privacy. Finally, we demonstrate how our predicate encryption for "small supersets" naturally captures hidden vector encryption.

Remark on Presentation. After the technical overview, we will not return to the construction of public-key anonymous IBE based on Canetti's obfuscator [Can97]; the construction described in this technical overview follows trivially from our full-fledged "small superset" obfuscator in Section 5. Details and defini-

[12] The "big subset" function of Beullens and Wee [BW19] is also parameterized by the same n, t, X, but it outputs 1 if and only if $Y \subseteq X$ and $|Y| \geq t$. The functionalities are seen to be equivalent by associating each input set Y with its complement $[n] \setminus Y$.

tions for our extensions to d-CNFs and read-once conjunctions of d-disjunctions (for constant d) can be found in the full version; we note that these constructions follow from a straightforward generalization of our main techniques.

Function-Private Anonymous IBE from Point Obfuscation We start with Canetti's point function obfuscator [Can97]. Recall that a point function I_x is a boolean-valued function that outputs 1 on input x, and 0 elsewhere. Fix a cryptographic group with order p and generator g. Given x, we obfuscate I_x by drawing a uniformly random $r \leftarrow \mathbb{F}_p$ and outputting

$$\mathsf{Obf}(I_x) = (g^r, g^{rx}).$$

Anyone can evaluate I_x on arbitrary input y by computing $(g^r)^y$ and comparing with g^{rx}. Moreover, Canetti proves that if x is drawn from any distribution with super-logarithmic min-entropy, the above construction hides x under a strengthening of the Decisional Diffie-Hellman (DDH) assumption [Can97].

Handling Encrypted Inputs: A First Attempt. A natural idea to upgrade Canetti's obfuscator to work for encrypted inputs y is to use a bilinear map, and to "obfuscate" the input y in a similar manner. Consider groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with associated generators g_1, g_2, g_T equipped with a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. To generate the public key, we draw a uniformly random $r \leftarrow \mathbb{F}_p$ and give out $g_2^{r^{-1}}$. We treat r as a secret key which is given to the obfuscator. To encrypt a plaintext y, the user computes $(g_2^{r^{-1}})^{y^{-1}}$. A function decryption key for I_x is simply $\mathsf{Obf}(I_x) = g_1^{rx}$.

A user holding an encryption $g_2^{r^{-1}y^{-1}}$ of y and a function decryption key g_1^{rx} for I_x can easily verify whether $I_x(y) = 1$ (i.e. $y = x$) by using the bilinear map and checking whether

$$e(g_1^{rx}, g_2^{r^{-1}y^{-1}}) \stackrel{?}{=} g_T.$$

However, this simple method of "encrypting" y fails to achieve even semantic security for ciphertexts since the encryption algorithm is deterministic. That is, an attacker trying to distinguish between an encryption of y_0 and an encryption of y_1 can easily encrypt both and compare to the challenge ciphertext. A natural approach to randomizing the encoding procedure would be to encode y as $g_2^{r^{-1}y^{-1}\alpha}$ for a random $\alpha \leftarrow \mathbb{F}_p$. However, for evaluation to work, the ciphertext would have to include g_T^{α}, and essentially the same limitation would arise since the attacker can request decryption keys of their choice.[13]

[13] In more detail, an attacker trying to distinguish between an encryption of y_0 and an encryption of y_1 (for y_0, y_1 of their choice) is free to request decryption keys corresponding to any function I_x provided that I_x does not trivially allow the attacker to distinguish between y_0 and y_1. The attacker can therefore request g_1^{rx} for any x that does not equal y_0 or y_1. Given challenge $g_2^{\alpha r^{-1} y_b^{-1}}, g_T^{\alpha}$ and decryption key g_1^{rx},

Handling Encrypted Inputs Securely. Our goal now is to modify the scheme so that we can introduce randomness into the encryption procedure without disturbing correctness. The idea is to generalize the above procedure to first encode x as a 2-dimensional vector $[x\ x^2]$, and to replace the role of y^{-1} with a uniformly random vector orthogonal to $[y\ y^2]$. Now if we compute the dot product of these vectors, we get 0 if $y = x$ and a non-zero value otherwise. Note that a random vector orthogonal to $[y\ y^2]$ can be written as $[-\beta y\ \beta]^\top$ where $\beta \leftarrow \mathbb{F}_p$ is uniformly random. The role of the random scalar r in the previous scheme can be replaced by a uniformly random invertible 2×2 matrix $\mathbf{R} \leftarrow \mathbb{F}_p^{2\times 2}$. We are also free to introduce independent randomness α during obfuscation/secret key generation. The resulting scheme is as follows.

– Setup. Draw random invertible $\mathbf{R} \leftarrow \mathbb{F}_p^{2\times 2}$, and output $\mathsf{pk} = g_2^{\mathbf{R}^{-1}}, \mathsf{sk} = \mathbf{R}$.[14]
– $\mathsf{KeyGen}(\mathsf{sk}, x)$. Parse sk as $\mathbf{R}$. Draw random $\alpha \leftarrow \mathbb{F}_p$ and output $\mathsf{sk}_x = g_1^{[\alpha x\ \alpha x^2]\mathbf{R}}$.
– $\mathsf{Enc}(\mathsf{pk}, y)$. Parse pk as $g_2^{\mathbf{R}^{-1}}$. Draw random $\beta \leftarrow \mathbb{F}_p$ and output $g_2^{\mathbf{R}^{-1}[-\beta y\ \beta]^\top}$.
– $\mathsf{Dec}(\mathsf{sk}_x, c)$. Parse sk_x as $g_1^{[v_1\ v_2]}$ and c as $g_2^{[u_1\ u_2]^\top}$. Use the bilinear map e to compute $g_T^{[v_1\ v_2]\cdot\left[\begin{smallmatrix}u_1\\u_2\end{smallmatrix}\right]}$ and output 1 if this equals g_T^0.

Adding Payloads for Function-Private Anonymous IBE. At the moment, the above scheme corresponds to an IBE scheme without message payloads; if we interpret x and y as user identities, currently a user only learns whether or not they were the correct recipient of a ciphertext. To obtain full IBE, we need to modify the encryption algorithm to incorporate a message payload μ. To enable this, we extend $\mathbf{R}$ to a 3×3 matrix, and extend the obfuscated row vector to $[1\ x\ x^2]$. During encryption we choose more randomness γ, extend the encrypted column vector to $[\gamma\ k_1\ k_2]^\top$, and additionally release $\mu \cdot g_T^\gamma$. An accepting input will now decrypt to g_T^γ rather than the identity, which can be divided out from $\mu \cdot g_T^\gamma$ to recover μ.

On Function-Private Identity Based Encryption. A construction of function-private anonymous IBE appears in Boneh, Raghunathan, and Segev [BRS13a]. Their approach starts with an existing (anonymous) IBE scheme and "upgrades" it to statistically hide the function using a randomness extractor. As outlined earlier, our approach and construction differ in several important dimensions. First, our approach starts with an existing point obfuscation scheme, "upgrades" it to encrypt the inputs, and then subsequently introduces the ability to encrypt a

the attacker can use the fact that they know x, y_0, y_1 in the clear to determine b as follows. The attacker raise g_T^α to the exponent xy_0^{-1} to obtain $g_T^{\alpha x y_0^{-1}}$, and then computes $e(g_1^{rx}, g_2^{\alpha r^{-1} y_b^{-1}})$. If $b = 0$, these quantities match, and otherwise they do not.

[14] We use the shorthand $g^{\mathbf{V}}$ where $\mathbf{V} = (v_{i,j})_{i\in[k],j\in[\ell]}$ to denote the matrix of group elements $(g^{v_{i,j}})_{i\in[k],j\in[\ell]}$.

message payload. Second, our construction achieves computational function privacy for any distribution with super-logarithmic min-entropy, rather than λ min-entropy as required in [BRS13a] (this requirement was also relaxed in [PMR19]). However, the drawback of our approach is that we can only prove security in the generic (bilinear) group model [Nec94, Sho97, Mau05], whereas [BRS13a] is proven secure in the standard model.

Building Public-Key Function-Private Predicate Encryption for "Small Supersets" We now briefly describe how to extend the above function-private anonymous IBE to handle the significantly more expressive "small superset" functionality, described earlier. First, we describe how to generate a function decryption key for $f_{n,t,X}$, where $X \subseteq [n]$. Now, $\mathbf{R}$ is a uniformly random width $t+1$ matrix (instead of width 2). We now follow essentially the same procedure as before for each $x \in X$. That is, for each $x \in X$ we form the row vector $[x\ x^2\ \ldots\ x^{t+1}]$ and compute the row vector $[x\ x^2\ \ldots\ x^{t+1}] \cdot \mathbf{R}$. We collect the row vectors resulting from this process into a matrix $\mathbf{M}_X$ where the rows are indexed by elements $x \in X$.

A set $Y \subseteq [n]$ (corresponding to a set that will be given as input to the "small superset" functionality) can be encrypted as follows. We assemble a matrix $\mathbf{W}_Y$ whose rows are indexed by elements $y \in Y$. The row corresponding to y is simply $[y\ y^2\ \ldots\ y^{t+1}]$. Draw a uniformly random vector $\mathbf{v}$ in the right kernel of $\mathbf{W}_Y$, and output $\mathbf{v}_Y = \mathbf{R}^{-1} \cdot \mathbf{v}$. Note that this is only possible if $|Y| \leq t$.

To decrypt, compute the matrix-vector product $\mathbf{M}_X \cdot \mathbf{v}_Y$ in the exponent, which will be the all-zeros vector if and only if $X \subseteq Y$. To minimize the size of the obfuscation, we can collapse the matrix $\mathbf{M}_X$ to a vector $\mathbf{u}_X^\top$, by left multiplying by a uniformly random vector of the appropriate dimension. Then decryption simply computes the (dot) product $\mathbf{u}_X^\top \cdot \mathbf{v}_Y$ in the exponent. In the body, we describe these obfuscation and encryption procedures in the language of linear codes, which results in a cleaner presentation.

Note that our construction is efficient as long as t is polynomial in the security parameter, since the vector and matrix dimensions are all determined by t. In particular, the universe size n could be exponential. On the other hand, the obfuscation construction given by Beullens and Wee [BW19] for large subset is only efficient for polynomial sized universe.

Finally, we remark that it is also easy to extend this to function-private predicate encryption for small superset by adding a payload in the same manner as for identity based encryption.

Function-Private Hidden Vector Encryption. We now describe how function-private predicate encryption for small superset gives rise to function-private hidden vector encryption [BW07]. Consider a vector $\mathbf{v} = (v_i)_{i \in [k]} \in (\mathbb{Z}_s \cup \{*\})^k$. Hidden vector encryption corresponds to predicate encryption for the predicate

$$P_{\mathbf{v}}(\mathbf{u}) = \begin{cases} 1 & \text{if for all } i \in [k]\text{: } (v_i = u_i \text{ or } v_i = *), \\ 0 & \text{otherwise.} \end{cases}$$

Let the universe size of the small superset instance be $n = ks$ and the threshold value be k. Let the set X corresponding to $\mathbf{v}$ be defined as $X \coloneqq \{(i-1)s+v_i\}_{i\in B}$, where B denotes the non-wildcard positions of $\mathbf{v}$. Then an input vector $\mathbf{u} = (u_i)_{i\in[k]} \in \mathbb{Z}_s^k$ corresponds to the set $Y_{\mathbf{u}} \coloneqq \{(i-1)s + u_i\}_{i\in[k]}$, which has size exactly k. Finally, we have $P_{\mathbf{v}}(\mathbf{u}) = 1 \iff X \subseteq Y_{\mathbf{u}}$. Since hidden vector encryption is most generally defined over exponentially sized alphabets, we would like to take s and thus n to be exponential. Thus, we crucially rely on the fact that the universe size of our small superset instance is allowed to be exponential.

Function-Privacy Definitions. When considering public-key function-private predicate encryption, the appropriate notion of function privacy is somewhat tricky to define. We choose to generalize the original notion of "real-or-random" function privacy of Boneh et al. [BRS13a]. This definition was originally stated just for point functions and was later extended to inner products [BRS13b]. Roughly, the definition considers an oracle which is set to be in either "real" or "random" mode, and which accepts a distribution over points. If it is in real mode, it produces a key for a point drawn from the queried distribution, and if it is in random mode, it produces a key for a uniformly random point. Security is parameterized by a class of allowed distributions for which the adversary can query its oracle, and requires that an adversary cannot determine which mode its oracle is in.

Extending this definition to a larger class $\mathcal{C}$ of functions would require a natural notion of a uniformly random function from $\mathcal{C}$. We choose to instead view the random mode as a "simulated" mode, where the behavior of the oracle is independent of the queried distribution, but otherwise arbitrary. This definition now naturally extends to any class of functions, and captures the same intuition that an adversary learns nothing about the function that it has a key for, as long as it is drawn from a particular class of distributions. We refer to this oracle now as the Real-or-Sim oracle.

We note here that although our predicate encryption constructions are inspired by and built from existing obfuscation constructions (in particular, those that already satisfy distributional virtual black box security), this notion of function privacy is incomparable to distributional VBB. In particular, distributional VBB is defined relative to a distribution $\mathcal{D}$ over functions in $\mathcal{C}$, and essentially requires that no adversary, given the obfuscation of a function f drawn from $\mathcal{D}$, can guess the value of any predicate $\mathcal{P}$ applied to f. On the other hand, our definition of function privacy is defined relative to an entire class of distributions $\mathbb{D}$, and does not consider predicates on functions drawn from individual distributions $\mathcal{D}$. Instead, we require that the class of distributions $\mathbb{D}$ is simulatable in the sense described above. Note that our constructions also satisfy distributional VBB, but we focus on this function-private predicate encryption style of definition, as it aligns more closely with previous work.

Enhanced Function Privacy. As in prior work [BRS13a], we will be concerned with evasive distributions over functions, where it is difficult to find an accepting input given oracle access to a function drawn from the distribution. However, it is crucial for applications that given a decryption key for an unknown function, the key can be used to successfully decrypt payloads without sacrificing function privacy. This means in particular that an adversary should not be able to produce accepting inputs to its decryption key, *even given* encryptions of arbitrary accepting inputs.

This is captured by Boneh et al. [BRS13a] by the notion of *enhanced* function privacy, where in the real-or-random game, the adversary is additionally given an encryption oracle. The adversary can query this oracle to obtain encryptions of arbitrary accepting inputs to the unknown functions corresponding to the decryption keys in its possession. Enhanced function privacy requires that the adversary still cannot determine what mode its Real-or-Sim oracle is in. We prove that our predicate encryption scheme for small superset satisfies this enhanced function privacy notion, which implies that our hidden vector encryption construction does as well.

Secure Distributions for Function-Private HVE (and More). We determine which distributions over HVE instances induce an evasive distribution over small superset instances, under the mapping defined above. We parameterize HVE distributions by an alphabet size s, and an input length k. For a particular distribution $\mathcal{D}_{k,s}$, let $H_\infty(\mathcal{D}_{k,s})$ be the min-entropy of the vector $\mathbf{v} \leftarrow \mathcal{D}_{k,s}$. Following the proof strategy from [BW19, Lemma 2], we show enhanced function-private hidden vector encryption for the set of distributions containing any $\mathcal{D}_{k,s}$ such that $H_\infty(\mathcal{D}_{k,s}) \geq k + \omega(\log k)$.

Note that the min-entropy requirement scales with the input length, but not with the alphabet size. Thus as the alphabet size increases, we obtain security for a larger and larger class of distributions. If we instead had a polynomial limit on the universe size of our small superset instances (like in [BW19]), then to support exponentially large alphabets $\mathbb{Z}_s$, we would be forced to first write each element as a bitstring (or more generally a string over a polynomially sized alphabet), increasing the input length. This would cause the min-entropy requirement to scale with the size of the alphabet.

On the other hand, this result severely restricts the possible distributions when s is a small constant. Thus we give an additional set of secure distributions (that also appear in [BW19] in the context of conjunction obfuscation) over vectors with a fixed number of wildcards w. We obtain enhanced function-private hidden vector encryption for the set of distributions containing any

$$\mathcal{D}_{k,s} \text{ such that } H_\infty(\mathcal{D}_{k,s}) = \log \binom{k}{w} + \omega(\log(k)),$$

and where $\mathcal{D}_{k,s}$ is supported on vectors with exactly w wildcards. Note that for some values of w, this min-entropy bound is much less than the input length k, and thus supports a large and interesting class of distributions even for small alphabet size s.

Extentions to d-CNF and Read-once Conjunction of d-disjunctions. We also extend the enhanced-function-private predicate encryption of "small supersets" to d-CNF and conjunction of d-disjunctions for $d = O(1)$.

d-CNFs for $d = O(1)$. The underlying technique in the BKMPRS construction is to translate the evaluation of the conjunction functionality into a polynomial interpolation, which is successful if and only if all input values (one per comparison clause) are valid points on the underling polynomial. This is achieved by evaluating the comparison functionality as a *lookup table* which contains either valid shares for matching input, or random values, otherwise (all encoded in the exponent for security). Our observation is that we can use a similar lookup table approach to implement *any circuit functionality* besides comparisons, and this technique is polynomially efficient as long as the underlying circuits have constant input length.

A d-CNF for k-bit input is a circuit $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where for each $i \in [m]$, C_i is a boolean circuit which depends only on the inputs bits with indices in a subset, denoted as $I_i \subseteq [k]$. We now show how to reduce the d-CNF to the "small superset" functionality.

Given a d-CNF $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, denote $K = \binom{k}{d}$ and $D = 2^d$. We create a universe of $n = KD$ elements. Then we reform the set $[KD]$ into a $K \times D$ matrix. The rows of the matrix corresponds to subsets of size d in $[k]$. The columns of the matrix corresponds to the input strings of k-bits. Now we specify a subset X of $[KD]$. For $I \in \binom{[k]}{d}$ and $v \in \{0,1\}^k$, X contains the elements in I-th row and v-th column, if there exists a $C_i, i \in [m]$ such that C_i only depends on I and $C_i(v) = 0$. On input $x \in \{0,1\}^k$, we specify a subset $Y \in [KD]$. For every $I \in \binom{[n]}{d}$, Y contains all elements in I-th row, except the one in x_I-th colum. Since Y contains $K(D-1)$ elements, we simply set the threshold $t = K(D-1)$. Then, $C(x) = 1$ if and only if $X \subseteq Y$. This is because, $C(x) = 1$ if and only if the following condition holds: for every $I \in \binom{[k]}{d}$, either there exists a C_i such that C_i only depends on I and $C_i(x_I) = 1$, or such C_i doesn't appear in C. The above condition is equivalent to $X \subseteq Y$. We prove security for some special distributions over d-CNF.

Function Distribution. We prove the security for two distributions. The first distribution essentially corresponds to the "uniform" case. Here, we achieve the *best possible* parameter, namely, $m = \omega(\log k)$.[15] Our proof in this case is a natural extension of the BKMPRS proof. The second distribution is useful for obtaining obfuscation of conjunctions of d-disjunctions via the mapping discussed earlier. Crucially, in this distribution, we do *not* require the distribution over C_i to be independent. Consequently, the proof of security for this distribution is more involved. Specifically, since C_i's may be dependent, in order to use a combinatorial argument similar to BKMPRS, we first need to "break" the dependence. We address this by choosing a subset of sets, say $\mathcal{I}$, such that the sets in $\mathcal{I}$ are disjoint. Clearly, $\mathcal{I}$ has the necessary independence. To choose such a subset, we build a graph, where each vertex of the graph represents a set, and draw an

[15] Indeed, m must be $\omega(\log k)$ in order to make the function family evasive.

edge between the two vertices if and only if the intersection of two vertex is non-empty. We then bound the degree of this graph and argue that the number of color used for coloring the graph is also bounded. Finally, we use the pigeonhole principle to pick such a subset $\mathcal{I}$. Due to lack of space, we refer the reader to the full version for details.

Read Once Conjunctions of d-Disjunctions. We also consider a class of functionalities that directly generalizes the conjunctions functionality but in a different way from d-CNFs. While the conjunctions functionality constrains the value of each input bit independently, in our generalization we constrain the values of several consecutive input bits together. More precisely, our functionality is defined as

$$\left(p_1^{(1)} \vee \cdots \vee p_d^{(1)}\right) \wedge \cdots \wedge \left(p_1^{(\ell)} \vee \cdots \vee p_d^{(\ell)}\right),$$

where $p_j^{(i)}$ is a length k_i string over alphabet $\{0, 1, ?\}$, and $\sum_i k_i = k$. It evaluates to one on input string $x = x^{(1)} \| \cdots \| x^{(\ell)} \in \{0,1\}^n$ if and only if for every $i \in [\ell]$, it holds that $|x^{(i)}| = k_i$ and $x^{(i)}$ matches one of $\{p_j^{(i)}\}_{j \in [d]}$.

One direct way to achieve the above functionality using the d-CNF construction is by considering each $\left(p_1^{(i)} \vee \cdots \vee p_d^{(i)}\right)$ as the functionality of the clause C_i. However, this will impose a restriction that each $k_i = O(1)$. Instead, We provide a different mapping to the d-CNF class with the only restriction that $\sum_i k_i = k$ when $k = O(1)$. This mapping transforms the conjunction of disjunction over strings into a conjunctions of disjunctions over bits by representing the matching $y \overset{?}{=} x$ of a longer string $x = x_1 \ldots x_t$ as the conjunction over bit comparisons $y_1 \overset{?}{=} x_1 \wedge \cdots \wedge y_t \overset{?}{=} x_t$.

1.3 Outline

The rest of the paper is structure as follows. In Section 2 we define notation and provide background definitions. In Section 3 we present our construction for obfuscating small supersets. In Section 4 we formally define our security notions of data privacy and (enhanced) function privacy, and in Section 5 we present our construction for function-private predicate encryption for small supersets. In the full version we present applications of our construction to hidden vector encryption, d-CNFs for $d = O(1)$, and read-once conjunctions of d-disjunctions for $d = O(1)$.

2 Preliminaries

We use the standard Landau notations. A function $\epsilon(\lambda)$ is written as $\mathsf{negl}(\lambda)$ if for all positive integers c, $\epsilon(\lambda) = o(\frac{1}{\lambda^c})$. For a positive integer n, we let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For a finite set S, $x \leftarrow S$ denotes a uniformly random sample.

We write scalars as lowercase unbolded letters (e.g., α or a), vectors as lowercase bold letters (e.g., $\mathbf{v}$) and matrices as uppercase bold letters (e.g. $\mathbf{M}$). We use the shorthand $g^{\mathbf{v}}$ where $\mathbf{v} = (v_1, \ldots, v_n)$ to denote the vector of group elements $g^{v_1}, \ldots, g^{v_n}$, and naturally extend this notation to matrices $\mathbf{V}$. To distinguish between the case where x refers to a specific value and the case where x is used as a formal variable, we will explicitly write $\hat{x}$ if it is a formal variable. This notation will also extend to vectors $\hat{\mathbf{v}} = (\hat{v}_1, \ldots, \hat{v}_n)$ where each entry is itself a formal variable, as well as to matrices $\hat{\mathbf{M}}$ where each entry is a formal variable.

2.1 Bilinear Groups

We briefly recall the definition of an asymmetric bilinear group [Jou04, BF01]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be distinct groups, all of prime order q, and let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ onto the target group $\mathbb{G}_T$. Let g_1, g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is an asymmetric bilinear group if the following conditions are met:

- (Efficiency) The group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as well as the mapping $e(\cdot, \cdot)$ are all efficiently computable.
- (Non-degeneracy) $e(g_1, g_2) = g_T$, where g_T is a generator of $\mathbb{G}_T$.
- (Bilinearity) $e(g_1^a, g_2^b) = g_T^{ab}$ for all $a, b \in \mathbb{Z}_q$.

2.2 Generic Bilinear Group Model

We use an extension of the generic group model [Nec94, Sho97] adapted to bilinear groups. The following definition is taken verbatim from [KLM+18].

Definition 1 (Generic Bilinear Group Oracle). *A generic bilinear group oracle is a stateful oracle* BG *that responds to queries as follows:*

- *On a query* $\mathsf{BG.Setup}(1^\lambda)$*, the oracle generates two fresh nonces* $\mathsf{pp}, \mathsf{sp} \leftarrow \{0,1\}^\lambda$ *and a prime* p*. It outputs* $(\mathsf{pp}, \mathsf{sp}, p)$*. It stores the generated values, initializes an empty table* $T \leftarrow \{\}$*, and sets the internal state so subsequent invocations of* $\mathsf{BG.Setup}$ *fail.*
- *On a query* $\mathsf{BG.Encode}(k, x, i)$ *where* $k \in \{0,1\}^\lambda, x \in \mathbb{Z}_p$ *and* $i \in \{1, 2, T\}$*, the oracle checks that* $k = \mathsf{sp}$ *(returning* $\perp$ *otherwise). The oracle then generates a fresh nonce* $h \leftarrow \{0,1\}^\lambda$*, adds the entry* $h \mapsto (x, i)$ *to the table* T*, and outputs* h*.*
- *On a query* $\mathsf{BG.Add}(k, h_1, h_2)$ *where* $k, h_1, h_2 \in \{0,1\}^\lambda$*, the oracle checks that (1)* $k = \mathsf{pp}$*, and (2) the handles* h_1, h_2 *are present in its internal table* T *and are mapped to the values* (x_1, i_1) *and* (x_2, i_2)*, respectively, with* $i_1 = i_2$ *(returning* $\perp$ *otherwise). The oracle then generates a fresh handle* $h \leftarrow \{0,1\}^\lambda$*, computes* $x = x_1 + x_2 \in \mathbb{Z}_p$*, adds the entry* $h \mapsto (x, i_1)$ *to* T*, and outputs* h*.*
- *On a query* $\mathsf{BG.Pair}(k, h_1, h_2)$ *where* $k, h_1, h_2 \in \{0,1\}^\lambda$*, the oracle checks that (1)* $k = \mathsf{pp}$*, and (2) the handles* h_1, h_2 *are present in* T *and are mapped to values* $(x_1, 1)$ *and* $(x_2, 2)$*, respectively (returning* $\perp$ *otherwise). The oracle then generates a fresh handle* $h \leftarrow \{0,1\}^\lambda$*, computes* $x = x_1 x_2 \in \mathbb{Z}_p$*, adds the entry* $h \mapsto (x, T)$ *to* T*, and outputs* h*.*
- *On a query* $\mathsf{BG.ZeroTest}(k, x)$ *where* $k, x \in \{0,1\}^\lambda$*, the oracle checks that (1)* $k = \mathsf{pp}$*, and (2) the handle* h *is present in* T *and it maps to some value* (x, i) *(returning* $\perp$ *otherwise). The oracle then outputs "zero" if* $x = 0 \in \mathbb{Z}_p$ *and "non-zero" otherwise.*

2.3 Virtual Black Box Obfuscation

We recall the definition of a distributional virtual black-box (VBB) obfuscator. We roughly follow the definition of Brakerski and Rothblum [BR13].

Definition 2 (Distributional VBB Obfuscation). *Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of polynomial-size circuits, where $\mathcal{C}_n$ is a set of boolean circuits operating on inputs of length n, and let* Obf *be a PPT algorithm which takes as input an input length $n \in \mathbb{N}$ and a circuit $C \in \mathcal{C}_n$ and outputs a boolean circuit $\mathsf{Obf}(C)$ (not necessarily in $\mathcal{C}$). Let $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ be an ensemble of distribution families $\mathcal{D}_n$ where each $D \in \mathcal{D}_n$ is a distribution over $\mathcal{C}_n$.*

Obf *is a distributional VBB obfuscator for the distribution class $\mathcal{D}$ over the circuit family $\mathcal{C}$ if it has the following properties:*

1. *(Strong) Functionality Preservation: For every $n \in \mathbb{N}$, $C \in \mathcal{C}_n$, there exists a negligible function μ such that*
$$\Pr[\mathsf{Obf}(C, 1^n)(x) = C(x)\ \forall x \in \{0,1\}^n] = 1 - \mu(n)\,.$$
2. *Polynomial Slowdown: For every $n \in \mathbb{N}$ and $C \in \mathcal{C}_n$, the evaluation of $\mathsf{Obf}(C, 1^n)$ can be performed in time poly$(|C|, n)$.*
3. *Distributional Virtual Black-Box: For every PPT adversary $\mathcal{A}$, there exists a (non-uniform) polynomial size simulator $\mathcal{S}$ such that for every $n \in \mathbb{N}$, every distribution $D \in \mathcal{D}_n$ (a distribution over $\mathcal{C}_n$), and every predicate $\mathcal{P} \colon \mathcal{C}_n \to \{0,1\}$, there exists a negligible function μ such that*
$$\left|\Pr_{C\leftarrow\mathcal{D}_n}[\mathcal{A}(\mathsf{Obf}(C,1^n)) = \mathcal{P}(C)] - \Pr_{C\leftarrow\mathcal{D}_n}[\mathcal{S}^{C}(1^{|C|}, 1^n) = \mathcal{P}(C)]\right| = \mu(n)\,.$$

2.4 Predicate Encryption

Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ be a function class, where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$. Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ be a message space.

Definition 3 (Public-key Predicate Encryption). *A* public-key predicate encryption *scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for a function class $\mathcal{F}$ and message space $\mathcal{M}$ is a tuple of* PPT *algorithms defined as follows:*

– $\mathsf{Setup}(1^\lambda)$*: On input security parameter $\lambda \in \mathbb{N}$ provided in unary, output master secret key* msk *and public key* pk.
– $\mathsf{KeyGen}(\mathsf{msk}, f)$*: On input master secret key* msk *and function $f \in \mathcal{F}_\lambda$, output decryption key* sk_f.
– $\mathsf{Enc}(\mathsf{pk}, x, \mu)$*: On input public key* pk *an attribute $x \in \mathcal{X}_\lambda$, and a payload $\mu \in \mathcal{M}_\lambda$, output ciphertext* ct.
– $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$*: On input decryption key* sk_f *for function $f \in \mathcal{F}_\lambda$ and ciphertext* ct*, output an element of $\mathcal{M}_\lambda \cup \{\perp\}$.*

A public-key predicate encryption scheme Π for function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$ is $\mathsf{correct}$ if for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda, x \in \mathcal{X}_\lambda$, and $\mu \in \mathcal{M}_\lambda$, it holds that:

$$\Pr\left[\begin{array}{l} (\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x, \mu) \end{array} \middle| \ \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = \begin{cases} \mu & \text{if } f(x) = 1 \\ \perp & \text{if } f(x) = 0 \end{cases}\right] = 1 - \nu(\lambda)\,,$$

where the probability is taken over the internal randomness of the algorithms and $\nu(\cdot)$ is a negligible function.

We defer the security notions for predicate encryption to Section 4.

3 Obfuscating Small Supersets

We define the "small superset" functionality. As mentioned earlier, this is an alternative but virtually identical view of the "big subset" functionality proposed by Beullens and Wee [BW19]. However, we find the "small superset" formulation to be significantly more intuitive for our applications. The small superset functionality $f_{n,t,X}$ is parameterized by a universe size n, a threshold value t, and a set $X \subseteq [n]$. $f_{n,t,X}$ takes as input a set $Y \subseteq [n]$, and accepts if $|Y| \leq t$ and $X \subseteq Y$. While Beullens and Wee [BW19] limit n to be polynomial-size, we integrate the approach of Bartusek, Lepoint, Ma, and Zhandry [BLMZ19] for large-alphabet conjunctions to handle exponential size n, provided $t = \mathsf{poly}(\lambda)$.

Definition 4. *Let $X \subseteq \mathbb{F}_q$ consist of elements $x_1, \ldots, x_k$. Let $\mathbf{B}_{t,X,q} \in \mathbb{F}_q^{k\times(t+1)}$ be defined as*

$$\mathbf{B}_{t,X,q} \coloneqq \begin{pmatrix} x_1 & x_1^2 & \ldots & x_1^{t+1} \\ x_2 & x_2^2 & \ldots & x_2^{t+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_k & x_k^2 & \cdots & x_k^{t+1} \end{pmatrix}.$$

We also define the following helper functionalities.

- $\mathsf{SampCodeword}(\mathbf{B} \in \mathbb{F}_q^{k\times(t+1)})$. Output a random codeword in the code generated by $\mathbf{B}$ by sampling uniformly random $\mathbf{e} \in \mathbb{F}_q^k$ and outputting $\mathbf{e}^\top \cdot \mathbf{B}$.
- $\mathsf{SampDualCodeword}(\mathbf{B} \in \mathbb{F}_q^{k\times(t+1)})$. Output a uniformly random vector $\mathbf{w} \in \mathbb{F}_q^{t+1}$ in the right kernel of $\mathbf{B}$, i.e., $\mathbf{w}$ such that $\mathbf{B} \cdot \mathbf{w} = 0$.

3.1 Small Superset Obfuscation Construction

In this section, we define a small superset obfuscator using the above helper functionalities. The following construction is similar to the generic group constructions in [BW19, BLMZ19] (which build on [BKM+18]), though the presentation is tailored to fit the scope of this work. We assume that global parameters (λ, n, t, q) are set in advance, where λ is the security parameter, $n = 2^{\mathsf{poly}(\lambda)}$ is the universe size, $t = \mathsf{poly}(\lambda)$ is the threshold size, and q is a prime larger than n (for strong functionality preservation, we will require that $q \geq 2^\lambda \binom{n'}{t} = 2^{\mathsf{poly}(\lambda)}$, where $n' = \max\{n, 2t\}$). Let $\mathbb{G}$ be a group of order q with generator g.

- $\mathsf{Obf}((n,t,q), X \subseteq [n])$. $\mathbf{c}^\top \leftarrow \mathsf{SampCodeword}(\mathbf{B}_{t,X,q})$. Output $g^{\mathbf{c}^\top}$ (interpreted as $g^{c_1}, \ldots, g^{c_{t+1}}$).
- $\mathsf{Eval}((n,t,q), g^{\mathbf{c}^\top} \in \mathbb{G}^{t+1}, Y \subseteq [n])$. Let $\mathbf{w} \leftarrow \mathsf{SampDualCodeword}(\mathbf{B}_{t,Y,q})$. Accept if and only if $g^{\mathbf{c}^\top \cdot \mathbf{w}} = g^0$.

3.2 Functionality Preservation

We rely on the following fact (also stated in [BLMZ19]).

Lemma 1. *For any $t+1$ values of $x_1, \ldots, x_{t+1} < q$, the corresponding set of $t+1$ vectors $\{(x_i \; x_i^2 \cdots x_i^{t+1})\}_{i \in [t+1]}$ are linearly independent over $\mathbb{F}_q$.*

Functionality preservation now follows almost immediately from the following.

Lemma 2. *Let $X, Y \subseteq \mathbb{Z}_q$ be such that $|X|, |Y| \leq t$ and $X \not\subseteq Y$. Let $\mathbf{c}^\top \leftarrow \mathsf{SampCodeword}(\mathbf{B}_{t,X,q})$ and $\mathbf{w} \leftarrow \mathsf{SampDualCodeword}(\mathbf{B}_{t,Y,q})$. Then $\Pr[\mathbf{c}^\top \cdot \mathbf{w} = 0] \leq 2/q$.*

Proof. We first show that $\Pr[\mathbf{B}_{t,X,q} \cdot \mathbf{w} = \mathbf{0}] \leq 1/q$. By definition, there must be some element $x \in X$ such that $x \notin Y$. By Lemma 1, the row vector $(x \; x^2 \; \ldots \; x^{t+1})$ is not in the row span of $\mathbf{B}_{t,Y,q}$. Thus, only a $1/q$ fraction of the vectors in the kernel of $\mathbf{B}_{t,Y,q}$ are orthogonal to $(x \; x^2 \; \ldots \; x^{t+1})$. Noting that $\mathbf{w}$ is a uniformly random vector in the kernel of $\mathbf{B}_{t,Y,q}$, and that $(x \; x^2 \; \ldots \; x^{t+1})$ is a row of $\mathbf{B}_{t,X,q}$ establishes the claim. Finally, note that if $\mathbf{B}_{t,X,q} \cdot \mathbf{w} \neq 0$, then the uniform randomness of the vector $\mathbf{c}$ chosen by $\mathsf{SampCodeword}$ implies that $\Pr[\mathbf{c}^\top \cdot \mathbf{w} = 0] = 1/q$. Thus $\Pr[\mathbf{c}^\top \cdot \mathbf{w} = 0] \leq 1/q + ((q-1)/q)(1/q) \leq 2/q$. □

Now, if $X \subseteq Y$, then $\mathbf{w}$ is in the kernel of $\mathbf{B}_{t,X,q}$, so $\mathbf{c}^\top \cdot \mathbf{w} = 0$ with probability 1. Otherwise, the above lemma shows that $\mathbf{c}^\top \cdot \mathbf{w} \neq 0$ except with probability $2/q$. Now let $q \geq 2^\lambda \binom{n'}{t}$, where $n' = \max\{n, 2t\}$, and consider all sets Y such that $|Y| \leq t$ and $X \not\subseteq Y$. The number of such sets is at most $\sum_{i \in [t]} i \binom{n}{i} < t \binom{n'}{t}$. A union bound shows that strong functionality is preserved except with probability at most $t/2^\lambda = \mathsf{negl}(\lambda)$.

3.3 Security

Definition 5. *Let $n(\cdot)$ and $t(\cdot)$ be functions of the security parameter. We say that a family of distributions $\{\mathcal{D}_{n,t,\lambda}\}_\lambda$ where each $\mathcal{D}_{n,t,\lambda}$ is a distribution over subsets $X \subseteq [n(\lambda)]$ such that $|X| \leq t(\lambda)$, is an evasive distribution for the small-superset functionality, if for all fixed $Y \subseteq [n(\lambda)], |Y| \leq t(\lambda)$,*

$$\Pr[X \subseteq Y \mid X \leftarrow \mathcal{D}_{n,t,\lambda}] = \mathsf{negl}(\lambda).$$

Theorem 1. *For any functions $n(\cdot)$ and $t(\cdot)$, and evasive family of distributions $\{\mathcal{D}_{n,t,\lambda}\}_\lambda$ for the small superset functionality, the above construction is a distributional-VBB secure obfuscator in the generic group model.*

The proof is similar to the generic group proofs given in prior work [BLMZ19, BW19].

4 Function-Private Predicate Encryption Security Definitions

4.1 Data Privacy

Our data privacy definition is standard and captures the property that an adversary should not be able to tell the difference between two encrypted attributes x_0 and x_1 or payloads μ_0 and μ_1, provided that it does not have decryption keys that allow it to distinguish trivially. We allow the adversary access to a key generation oracle, allowing it to produce decryption keys for functions f of its choice (from a specified function class), subject to the usual requirement that $f(x_0) = f(x_1)$, and if $f(x_0) = f(x_1) = 1$ for some queried f, then $\mu_0 = \mu_1$.

Definition 6 (Data Privacy). *Let Π be a public-key predicate encryption scheme for function class $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ where $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \to \{0,1\}\}$ and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_\lambda$, and let $\mathcal{A}$ be a stateful adversary. We define the data privacy (DP) advantage as*

$$\mathbf{Adv}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda, 1) = 1\right] \right|,$$

where for $\lambda \in \mathbb{N}$ and $b \in \{0,1\}$, we define experiment $\mathsf{Expt}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda, b)$ as on the right, where $x_0, x_1 \in \mathcal{X}_\lambda$, and $\mu_0, \mu_1 \in \mathcal{M}_\lambda$. We additionally require that $\mathcal{A}$ is admissible in the following sense: for all KeyGen queries $f \in \mathcal{F}_\lambda$ made by $\mathcal{A}$ we have that $f(x_0) = f(x_1)$, and if there exists an f such that $f(x_0) = f(x_1) = 1$, then $\mu_0 = \mu_1$.

```
$\mathsf{Expt}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda, n, b)$

$(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$(x_0, x_1, \mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pk})$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b, \mu_b)$
**return** $\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{ct})$
```

We say Π is a data-private predicate encryption scheme if for all admissible PPT adversaries $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathbf{Adv}^{\mathsf{DP}}_{\Pi,\mathcal{A}}(\lambda) \leq \nu(\lambda)$.

4.2 Function Privacy

Now consider a set of distribution ensembles over functions, where for each choice of λ, we have a set of distributions $\mathbb{D}_\lambda$. Our first function privacy notion states that function keys for functions drawn from any distribution $\mathcal{D} \in \mathbb{D}_\lambda$ can be simulated even without the description of $\mathcal{D}$. We consider two experiments. In the first, the adversary has access to a "distributional key generation oracle" that takes as input some $\mathcal{D} \in \mathbb{D}_\lambda$ and outputs a decryption key for a boolean function f drawn from $\mathcal{D}$. In the second, we replace the distributional key generation oracle by a simulator. This simulator has no access to the input $\mathcal{D}$, and thus must produce "fake" decryption keys that are indistinguishable to any PPT adversary from real decryption keys produced by the key generation oracle.

Since we consider public-key schemes, an adversary essentially has oracle access to the function corresponding to any decryption key in its possession. Thus it may be easy for the adversary to distinguish these two experiments if for $f \leftarrow \mathcal{D}$, it can find an attribute x such that $f(x) = 1$. So this notion is only realizable for carefully chosen sets of distributions $\mathbb{D}_\lambda$, which in particular must consist solely of *evasive* distributions $\mathcal{D}$ [BBC+14]. That is, for $f \leftarrow \mathcal{D}$, finding x such that $f(x) = 1$ given oracle access to f is computationally intractable.

Definition 7 (Function Privacy). *Let Π be a public-key predicate encryption scheme for function class $\mathcal{F}$ and message space $\mathcal{M}$, let $\mathcal{A}$ be a stateful adversary, and let $\mathcal{S}$ be an explicit* PPT *algorithm simulating* KeyGen. *We define the function privacy (FP) advantage for set of distribution ensembles $\mathbb{D} = \{\mathbb{D}_\lambda\}_\lambda$ as*

$$\mathbf{Adv}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 1) = 1\right] \right|,$$

where for $\lambda \in \mathbb{N}$ and $b \in \{0, 1\}$, we define experiment $\mathsf{Expt}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, b)$ as:

```
$\mathsf{Expt}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 0)$

$(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
**return** $\mathcal{A}^{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{pk})$

$\underline{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk}, \mathcal{D})}$
If $\mathcal{D} \in \mathbb{D}_\lambda$, $f \leftarrow \mathcal{D}$, **return** $\mathsf{KeyGen}(\mathsf{msk}, f)$
Else **return** $\perp$
```

```
$\mathsf{Expt}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 1)$

$(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
**return** $\mathcal{A}^{\mathcal{S}(\mathsf{msk})}(1^\lambda, \mathsf{pk})$
```

We say Π is a $\mathbb{D}$-function-private predicate encryption scheme if there exists a simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathbf{Adv}^{\mathsf{FH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}) \leq \nu(\lambda)$.

4.3 Enhanced Function Privacy

In the standard notion of function privacy described above, the fact that the adversary only receives decryption keys sk_{f_j} for functions f_j (where f_j denotes the jth output of $\mathcal{O}_{\mathsf{DKeyGen}}$ during the course of the experiment) drawn from an evasive distribution $\mathcal{D}$ implies it will not be able to generate a ciphertext c encrypting (x, μ) such that $\mathsf{Dec}(\mathsf{sk}_{f_j}, c) \to \mu$ (except with negligible probability). We now describe a strictly stronger notion of function privacy known as *enhanced function privacy*, where we provide the adversary with an oracle $\mathcal{O}_{\mathsf{Enc}}$ that generates ciphertexts of (x, μ) such that that $f_j(x) = 1$ for some f_j. More precisely, $\mathcal{O}_{\mathsf{Enc}}$ takes pk and an index j as input, and outputs a ciphertext c of some arbitrary (x, μ) such that $\mathsf{Dec}(\mathsf{sk}_{f_j}, c) \to \mu$.

Note that normal (non-enhanced) function privacy does not guarantee any security the moment a function decryption key holder receives a ciphertext of (x, μ) such that $f(x) = 1$. This renders the standard function privacy notion almost useless in many settings, since as soon as a user is able to use its decryption key to decrypt any payload μ, all function privacy may be lost. We give our formal definition below, which generalizes the enhanced function privacy notion proposed by Boneh et al. [BRS13a, §3.2] in the context of identity-based encryption (IBE).

Definition 8 (Enhanced Function Privacy). *Let Π be a public-key predicate encryption scheme for function class $\mathcal{F}$ and message space $\mathcal{M}$, let $\mathcal{A}$ be a stateful adversary, and let $\mathcal{S} = (\mathcal{S}_{\mathsf{DKeyGen}}, \mathcal{S}_{\mathsf{Enc}})$ be an explicit* PPT *algorithm simulating* KeyGen *and* Enc. *We define the* enhanced function privacy (eFP) *advantage for distribution ensemble $\mathbb{D} = \{\mathbb{D}_\lambda\}_\lambda$ as*

$$\mathbf{Adv}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}) \overset{\text{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 1) = 1\right] \right|,$$

where for $\lambda \in \mathbb{N}$ and $b \in \{0, 1\}$, we define experiment $\mathsf{Expt}^{\mathsf{eFH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, b)$ as:

```
$\mathsf{Expt}^{\mathsf{eFH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 0)$

$(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
$j := 1$
return $\mathcal{A}^{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk},\cdot),\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk},\cdot)}(1^\lambda, \mathsf{pk})$

$\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk}, \mathcal{D})$
If $\mathcal{D} \in \mathbb{D}_\lambda$, $f_j \leftarrow \mathcal{D}$, return $\mathsf{KeyGen}(\mathsf{msk}, f)$
Else return $\perp$
$(j := j + 1)$

$\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk}, j)$
choose any $(x, \mu) \in \mathcal{X}_\lambda \times \mathcal{M}_\lambda$
  such that $f_j(x) = 1$
return $\mathsf{Enc}(\mathsf{pk}, x, \mu)$
```

```
$\mathsf{Expt}^{\mathsf{eFH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}, 1)$

$(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda)$
return $\mathcal{A}^{\mathcal{S}_{\mathsf{DKeyGen}}(\mathsf{msk}),\mathcal{S}_{\mathsf{Enc}}(\mathsf{pk},\cdot)}(1^\lambda, \mathsf{pk})$
```

We say Π is a $\mathbb{D}$-enhanced function-private predicate encryption scheme if there exists a simulator $\mathcal{S} = (\mathcal{S}_{\mathsf{DKeyGen}}, \mathcal{S}_{\mathsf{Enc}})$ such that for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathbf{Adv}^{\mathsf{eFH}}_{\Pi,\mathcal{S},\mathcal{A}}(\lambda, \mathbb{D}) \le \nu(\lambda)$.

4.4 Discussion

We view our enhanced function privacy definition as a direct generalization of the "real-or-random" enhanced function privacy definition considered by Boneh et al. [BRS13a]. Boneh et al. give their definition in the context of identity-based encryption (IBE), where an adversary is given an oracle that accepts distributions $\mathcal{D}$ over identities. The guarantee is that the adversary cannot determine whether the oracle is in "real" or "random" mode, where real mode means that it will return the secret key for an identity $\mathcal{I}$ drawn from the input distribution $\mathcal{D}$, and random mode means that it will return the secret key for a uniformly random identity. When attempting to generalize this definition to more expressive function classes (note that IBE corresponds to predicate encryption for point functions), it is not necessarily clear what the behavior of the random mode oracle should be.

We instead view the random mode oracle as a simulator which does not get to see the input distribution $\mathcal{D}$. In the case of IBE, one possible simulator could be defined to return a secret key for a uniformly random identity. But in general, we can allow the simulator's behavior to be arbitrary, as long as it does not depend on the queried distribution.

We note that our definition is weaker than the Boneh et al. definition [BRS13a] for IBE in one sense: we no longer provide the adversary with an explicit KeyGen oracle, which can be used to obtain secret keys for arbitrary functions of the adversary's choice (our only key generation oracle outputs functions drawn from evasive distributions). This is because such a definition is trivially unachievable when considering general functionalities such as small superset.

Indeed, since the behavior of the Enc oracle is arbitrary, assume that given an index j corresponding to a secret key for hidden subset X_j, it encrypts using the attribute X_j itself, a valid accepting input. Assume further that the universe size n is polynomial. Now an adversary can use the KeyGen oracle n times to receive a secret key for each of the subsets $\{i\}$ for $i \in [n]$. Then it simply tries to decrypt the encryption with attribute X_j with each of the keys, and can figure out exactly what X_j is, breaking function privacy. This style of attack does not exist when considering IBE where the functions encrypted are simply point functions.

As a final note about our definitions, we compare to those of Patranabis et al. [PMR19], which to the best of our knowledge is the only previous work proposing function private hidden vector encryption. They consider a notion of "left-or-right" security, where the adversary queries two distributions at a time to its oracle, and the oracle chooses which one to draw the function from depending on whether it is in "left" or "right" mode. They do not consider the enhanced version where an Enc oracle is provided, but they do provide a KeyGen oracle as in the original Boneh et al. [BRS13a] definition.

In the full version, we augment our basic (non-enhanced) function privacy definition to include a KeyGen oracle, sketch a proof that our small superset construction obtains this definition, and then show that this definition implies the left-or-right definition considered by Patranabis et al. [PMR19]. By going through our HVE-to-small-superset compiler and comparing the class of distributions considered in this work with those of Patranabis et al. [PMR19] (which, in particular, reveal the positions of the wildcards), we demonstrate that the security of our function private HVE construction generalizes that of Patranabis et al. [PMR19].

5 Function-Private Predicate Encryption for Small Superset

The following construction Π relies on an asymmetric bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We let $[a]_1, [b]_2, [c]_T$ denote encodings of a, b, c in groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively. For a vector $\mathbf{v}$ or matrix $\mathbf{M}$, we use the shorthand $[\mathbf{v}]$ or $[\mathbf{M}]$ (for any of the three groups) to denote the group elements obtained by encoding each entry of $\mathbf{v}$ or $\mathbf{M}$ respectively. Let $n(\cdot), t(\cdot)$ be functions of the security parameter. Let the message space $\mathcal{M} := \mathcal{M}_\lambda$ be a subset of the target group $\mathbb{G}_T$ such that $|\mathcal{M}|/|\mathbb{G}_T| = \mathsf{negl}(\lambda)$.[16]

– $\mathsf{Setup}(1^\lambda)$. Set $n := n(\lambda), t := t(\lambda)$. Pick a prime $q > \max\{n, 2^\lambda\}$. Sample a uniformly random matrix $\mathbf{R} \in \mathbb{F}_q^{(t+2)\times(t+2)}$ and compute $\mathbf{R}^{-1}$. Output $\mathsf{msk} := \mathbf{R}^{-1}$ and $\mathsf{pk} := [\mathbf{R}]_2$.
– $\mathsf{KeyGen}(\mathsf{msk}, X)$. Parse msk as $\mathbf{R}^{-1}$. To encrypt a subset $X \subseteq [n]$, draw $\mathbf{c}^\top \leftarrow \mathsf{SampCodeword}(\mathbf{B}_{t,X,q})$, sample $\alpha \leftarrow \mathbb{F}_q$, and output
$$[(1 \mid \alpha \cdot \mathbf{c}^\top) \cdot \mathbf{R}^{-1}]_1.$$
– $\mathsf{Enc}(\mathsf{pk}, Y, \mu)$. Parse pk as $[\mathbf{R}]_2$. To encrypt a message $Y \subseteq [n]$ such that $|Y| \le t$, let $\mathbf{w} \leftarrow \mathsf{SampDualCodeword}(\mathbf{B}_{t,Y,q})$, sample $\beta, \gamma \leftarrow \mathbb{F}_q$, and output
$$[\mathbf{R}]_2 \cdot (\gamma \mid \mathbf{w}^\top \cdot \beta)^\top, \quad \mu \cdot [\gamma]_T.$$
– $\mathsf{Dec}(\mathsf{sk}_X, \mathsf{ctxt}_Y)$. Parse sk_X as $[\mathbf{v}^\top]_1$ and ctxt_Y as $([\mathbf{w}]_2, h)$. Compute
$$\mu := h/[\mathbf{v}^\top \cdot \mathbf{w}]_T,$$
where the dot product in the target group is computed using the bilinear operation. If $\mu \in \mathcal{M}$, output μ, otherwise output $\perp$.

Correctness. If $X \subseteq Y$, note that the $\mathbf{w}$ vector associated with ctxt_Y is a codeword in the dual of the code from which the $\mathbf{c}^\top$ vector from sk_X was drawn. This follows since every row in the generator matrix of $\mathbf{c}^\top$'s code is one of the rows in the generator matrix of $\mathbf{w}$'s dual code. Thus the dot product computed during decryption will be equal to γ, and dividing h by $[\gamma]_T$ will give the encrypted payload μ.

If $X \not\subseteq Y$, a straightforward application of Lemma 2 shows that with overwhelming probability, $h/[\mathbf{v}^\top \cdot \mathbf{w}]_T$ will be a uniformly random group element, and will thus be an element of $\mathcal{M}$ with negligible probability. Thus, decryption will output $\perp$ with overwhelming probability.

[16] In [BW07], it is noted that this restriction on the size of the message space can be avoided in practice by essentially setting the payload to be the key of a symmetric key encryption scheme, and releasing an encryption of the actual message under this key (along with a consistency check). This technique can easily be applied in our setting.

5.1 Security

We make use of a variant [BGMZ18] of a lemma by Badrinarayanan et al. [BMSZ16]. In fact, we only need a particular special case of the lemma, stated below.

Lemma 3 ([BMSZ16, BGMZ18]). *Let $\hat{\mathbf{R}}$ be an $n \times n$ matrix of distinct formal variables $\hat{r}_{i,j}$, and $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$ be two arbitrary vectors. Let $\hat{\mathbf{u}} = \mathbf{u}^\top \cdot \hat{\mathbf{R}}^{-1}$ and $\hat{\mathbf{v}} = \hat{\mathbf{R}} \cdot \mathbf{v}$ be two vectors of rational functions over the $\hat{r}_{i,j}$ formal variables. Let P be a polynomial over the entries of $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ such that each monomial contains exactly one entry from $\hat{\mathbf{u}}$ and one from $\hat{\mathbf{v}}$. Then if P is identically a constant over the $\hat{r}_{i,j}$ variables, it must be a constant multiple of the inner product of $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$.*

Theorem 2. *The above construction Π is a* data-private *predicate encryption scheme for small superset.*

Proof. Consider any $Y_0, Y_1 \subseteq [n]$ such that $|Y_0|, |Y_1| \leq t$, and $\mu_0, \mu_1 \in \mathbb{G}_T$. The adversary $\mathcal{A}$ receives the public key and an encryption of (Y_b, μ_b) where $b \leftarrow \{0,1\}$. For convenience, we will let μ_0' and μ_1' be the discrete logs of μ_0, μ_1. $\mathcal{A}$ is free to request keys for sets X_i such that X_i is not contained in either Y_0 or Y_1. If $\mu_0 = \mu_1$, it is also free to request keys for sets X_i such that X_i is contained in both Y_0 and Y_1.

Thus, $\mathcal{A}$ has access to the handles of the elements

$$[\mathbf{R}]_2, \{[(1 \mid \alpha_i \cdot \mathbf{c}_{X_i}^\top) \cdot \mathbf{R}^{-1}]_1\}_i, [\mathbf{R} \cdot (\gamma \mid \mathbf{w}_{Y_b}^\top \cdot \beta)^\top]_2, [\mu_b' + \gamma]_T.$$

Recall that the only distinguishing information the adversary can obtain in the generic group model is the responses to zero-test queries in the target group. We first imagine replacing $\{\alpha_i\}_i, \beta, \gamma$, and the entries of $\mathbf{R}$ with formal variables. So we let $\hat{\mathbf{R}}$ be a $(t+2) \times (t+2)$ matrix of formal variables $\hat{r}_{i,j}$ for $i, j \in [t+2]$. We would like to apply Schwartz-Zippel to every zero-test query submitted by $\mathcal{A}$ in order to conclude that $\mathcal{A}$ cannot distinguish this switch except with negligible probability. However, the resulting zero-test expressions are rational functions of the above formal variables. We instead imagine taking each zero-test query and multiplying through by $\det(\hat{\mathbf{R}})$, which does not change whether it is identically zero or not. By construction, this results in a polynomial of degree at most $t + 5 = \mathsf{poly}(\lambda)$ over the formal variables. Thus applying Schwartz-Zippel and union bounding over the polynomially many zero-test queries submitted by $\mathcal{A}$ establishes that $\mathcal{A}$ cannot distinguish this switch except with negligible probability.

Now, define $(\hat{\mathbf{u}}^{(i)})^\top := (\hat{\alpha}_i^{-1} \mid \mathbf{c}_{X_i}^\top) \cdot \hat{\mathbf{R}}^{-1}$, and $\hat{\mathbf{v}} := \hat{\mathbf{R}} \cdot (\hat{\beta}^{-1}\hat{\gamma} \mid \mathbf{w}_{Y_b}^\top)^\top$. Using this notation, we will write down a general expression for any zero-test query submitted by the adversary. We consider all the possible ways that $\mathcal{A}$ can produce elements in the target group: pairing its ciphertext, secret key, or public key elements with a constant in the other group, or pairing its secret key elements with public key or ciphertext elements. Then we write a general linear combination of such elements, where $\kappa_{i,j}$, τ_k, $\delta_{k,\ell}$, $\eta_{i,j,k}$, $\rho_{i,j,k,\ell}$, and ν represent coefficients submitted by $\mathcal{A}$. This results in the following expression.

$$\begin{aligned}
&\sum_{i,j} \kappa_{i,j}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_k \tau_k \hat{\mathbf{v}}_k \hat{\beta} + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} + \sum_{i,j,k} \eta_{i,j,k}\hat{\alpha}_i (\hat{\mathbf{u}}_j^{(i)})^\top \hat{\mathbf{v}}_k \hat{\beta} \\
&\quad + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell} + \nu(\mu_b' + \hat{\gamma}) \\
&= \hat{\beta}\left(\sum_k \tau_k \hat{\mathbf{v}}_k + \sum_i \hat{\alpha}_i \left(\sum_{j,k} \eta_{i,j,k} (\hat{\mathbf{u}}_j^{(i)})^\top \hat{\mathbf{v}}_k\right)\right) + \sum_{i,j} \kappa_{i,j}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} \\
&\quad + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell} + \nu(\mu_b' + \hat{\gamma})
\end{aligned}$$

Now any potentially distinguishing zero-test query must result in an identically zero rational function for at least one setting of $b \in \{0,1\}$, and thus must set the coefficient on $\hat{\beta}$ to some scaling of $\hat{\beta}^{-1}$ for one of these settings (since $\hat{\beta}$ does not appear in the other terms). This implies a few things about the adversary's coefficients. First, for each k, $\tau_k = 0$, since each entry of $\hat{\mathbf{v}}$ is a sum over distinct formal variables from $\hat{\mathbf{R}}$ which cannot be canceled out elsewhere in the coefficient on $\hat{\beta}$. Next, for each i, the coefficient on $\hat{\alpha}_i$ within this $\hat{\beta}$ coefficient must be some scaling of $\hat{\alpha}_i^{-1}$. Then by Lemma 3, for each i, the coefficients $\{\eta_{i,j,k}\}_{j,k}$ must be set to induce a scaling of the inner product of $\hat{\mathbf{u}}^{(i)}$ and $\hat{\mathbf{v}}$. Let z_i denote this scaling. We can rewrite the above expression as follows.

$$\begin{aligned}
&\hat{\beta}\left(\sum_i \hat{\alpha}_i \left(z_i(\hat{\alpha}_i^{-1}\hat{\beta}^{-1}\hat{\gamma} + \mathbf{c}_{X_i}^\top \cdot \mathbf{w}_{Y_b})\right)\right) + \sum_{i,j} \kappa_{i,j}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} \\
&\quad + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell} + \nu(\mu_b' + \hat{\gamma}) \\
&= \hat{\gamma}\left(\sum_i z_i + \nu\right) + \hat{\beta}\left(\sum_i \hat{\alpha}_i z_i \mathbf{c}_{X_i}^\top \cdot \mathbf{w}_{Y_b}\right) + \sum_{i,j} \kappa_{i,j}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} \\
&\quad + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell} + \nu\mu_b'
\end{aligned}$$

Now observe that we need the coefficient on $\hat{\gamma}$ to be zero in order to obtain a successful zero-test. We consider two cases. First, if $z_i = 0$ for all i, then the coefficient on $\hat{\gamma}$ is zero only if $\nu = 0$. But in this case, the remaining term is

$$\sum_{i,j} \kappa_{i,j}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell}\hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell},$$

which is independent of the bit b. Thus, such a zero-test cannot be used to distinguish.

Otherwise, let S be the set of i such that $z_i \neq 0$. If the coefficient on $\hat{\beta}$ is zero for some b, this implies that $\mathbf{c}_{X_i}^{\top} \cdot \mathbf{w}_{Y_b} = 0$ for each $i \in S$, and thus by correctness, $X_i \subseteq Y_b$ for each $i \in S$. Then by admissibility, $X_i \subseteq Y_0, Y_1$ for each $i \in S$, meaning that the coefficient on $\hat{\beta}$ is zero regardless of b. But again by admissibility, this also implies that $\mu_0' = \mu_1'$. Then it is clear that the remaining expression

$$\sum_{i,j} \kappa_{i,j} \hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} + \sum_{k,\ell} \delta_{k,\ell} \hat{r}_{k,\ell} + \sum_{i,j,k,\ell} \rho_{i,j,k,\ell} \hat{\alpha}_i \hat{\mathbf{u}}_j^{(i)} \hat{r}_{k,\ell} + \nu \mu_b'$$

is independent of the bit b, completing the proof. □

Definition 9. *Let $n(\cdot), t(\cdot)$ be functions of the security parameter. Let $\mathcal{E}_{n,t}$ be the entire set of families of evasive small superset distributions $\{\mathcal{D}_{n,t,\lambda}\}_\lambda$. Write $\mathcal{E}_{n,t} = \{\mathcal{E}_{n,t,\lambda}\}_\lambda$.*

Theorem 3. *For any $n(\cdot), t(\cdot)$, the above construction Π is an $\mathcal{E}_{n,t}$-enhanced function-private predicate encryption scheme for small superset.*

Proof. In $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 0)$ from Definition 8, $\mathcal{A}$ interacts with an honest implementation of the construction Π in the generic bilinear group model. We prove through a series of hybrid experiments that $\mathcal{A}$'s view in the honest world is indistinguishable from its view in $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 1)$, in which the oracles $\mathcal{O}_{\mathsf{DKeyGen}}$ and $\mathcal{O}_{\mathsf{Enc}}$ are implemented by the simulator $\mathcal{S}$ with no knowledge of the queried distributions in $\mathcal{E}_{n,t,\lambda}$. Note that the oracles $\mathcal{O}_{\mathsf{DKeyGen}}$ and $\mathcal{O}_{\mathsf{Enc}}$ are allowed to share state.

First, we make explicit the following generic group instantiation of $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 0)$. Note that the adversary $\mathcal{A}$ in the below experiment and all following hybrid experiments also implicitly has access to generic group bilinear map operations described in Definition 1. Since $\mathcal{A}$ is PPT, we'll say that $\mathcal{A}$ makes $J = \mathsf{poly}(\lambda)$ queries to $\mathcal{O}_{\mathsf{DKeyGen}}$ and $K = \mathsf{poly}(\lambda)$ queries to $\mathcal{O}_{\mathsf{Enc}}$.

$\underline{\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 0)}$:

1. Set $n := n(\lambda), t := t(\lambda)$, $q > \max\{n, 2^\lambda\}$.
2. Sample $\mathbf{R} \leftarrow \mathbb{F}_q^{(t+2)\times(t+2)}$ and set $\mathsf{msk} := \mathbf{R}^{-1}$.
3. Generate fresh handles in group 2 for each entry of $\mathbf{R}$, letting pk consist of this set of handles.
4. Output $\mathcal{A}^{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk},\cdot), \mathcal{O}_{\mathsf{Enc}}(\mathsf{pk},\cdot)}(1^\lambda, \mathsf{pk})$.

$\underline{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk}, \mathcal{D})}$:

This oracle maintains an internal counter j, initialized at $j = 1$. After each oracle call, increment j. On each oracle call:

1. Sample $X_j \leftarrow \mathcal{D}$ and set $(\mathbf{c}^{(j)})^\top \leftarrow \mathsf{SampCodeword}(\mathbf{B}_{t,X_j,q})$.
2. Sample $\alpha_j \leftarrow \mathbb{F}_q$.
3. Set $(\mathbf{u}^{(j)})^\top := (1 \mid \alpha_j \cdot (\mathbf{c}^{(j)})^\top) \cdot \mathbf{R}^{-1}$.
4. Generate and return fresh handles in group 1 for $(\mathbf{u}^{(j)})^\top$.

$\underline{\mathcal{O}_{\mathsf{Enc}}(\mathsf{msk}, j)}$:
On the kth oracle call, do the following:

1. Let $Y_k \subseteq [n]$ be any set satisfying $|Y_k| \leq t$ and $X_j \subseteq Y_k$.
2. Let $\mu'_k \in \mathbb{F}_q$.
3. Sample:
   – $\mathbf{w}^{(k)} \leftarrow \mathsf{SampDualCodeword}(\mathbf{B}_{t,Y_k,q})$
   – $\beta_k, \gamma_k \leftarrow \mathbb{F}_q$
   – $\mathbf{v}^{(k)} := \mathbf{R} \cdot (\gamma_k \mid (\mathbf{w}^{(k)})^\top \cdot \beta_k)^\top$
4. Generate and return fresh handles in group 2 for $\mathbf{v}^{(k)}$, and a fresh handle in group T for $\mu'_k + \gamma_k$.

Now, we present a series of hybrid experiments, beginning with the above experiment and ending with a generic group instantiation of $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 1)$.

– Expt_0 is exactly $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{t,n}, 0)$.
– Expt_1 is obtained from Expt_0 by modifying $\mathcal{O}_{\mathsf{Enc}}(\mathsf{msk}, \cdot)$ to the following:
  $\underline{\mathcal{O}_{\mathsf{Enc}}(\mathsf{msk}, j)}$:
  On the kth oracle call, do the following:
  1. Let $\mu'_k \in \mathbb{F}_q$.
  2. Sample $\beta_k, \gamma_k \leftarrow \mathbb{F}_q$.
  3. Define t new formal variables $\hat{w}_{k,1}, \ldots, \hat{w}_{k,t}$.
  4. Define $\mathbf{w}^{(k)} := \left[\hat{w}_{k,1}, \ldots, \hat{w}_{k,t} - \frac{1}{\mathbf{c}^{(j)}_{t+1}} \sum_{i=1}^{t} \mathbf{c}^{(j)}_i \hat{w}_{k,i}\right]$.
  5. Set $\hat{\mathbf{v}}^{(k)} := \mathbf{R} \cdot (\gamma_k \mid (\mathbf{w}^{(k)})^\top \cdot \beta_k)^\top$.
  6. Generate and return fresh handles in group 2 for $\hat{\mathbf{v}}^{(k)}$, and a fresh handle in group T for $\mu'_k + \gamma_k$.

  Note that the generic bilinear group operations are now performed over the ring $\mathbb{Z}[\{\hat{w}_{k,i}\}_{k \in [K], i \in [t]}]$. Also note that $\mathcal{O}_{\mathsf{Enc}}$ and $\mathcal{O}_{\mathsf{DKeyGen}}$ are sharing state, in particular the set of $\mathbf{c}^{(j)}$ vectors.
– $\mathsf{Expt}_{2,\ell}$ (for $\ell = 0, \ldots, J$) is obtained from Expt_1, except $\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk}, \cdot)$ is modified to the following:
  $\underline{\mathcal{O}_{\mathsf{DKeyGen}}(\mathsf{msk}, \mathcal{D})}$:
  The oracle maintains an internal counter j, initialized at $j = 1$. After each oracle call, increment j. On each oracle call:
  1. If $j \leq \ell$, sample uniformly random $(\mathbf{c}^{(j)})^\top \leftarrow \mathbb{F}_q^{t+1}$. If $j > \ell$, sample $X_j \leftarrow \mathcal{D}$ and set $(\mathbf{c}^{(j)})^\top \leftarrow \mathsf{SampCodeword}(\mathbf{B}_{t,X_j,q})$.
  2. Sample $\alpha_j \leftarrow \mathbb{F}_q$.
  3. Set $(\hat{\mathbf{u}}^{(j)})^\top := (1 \mid \alpha_j \cdot (\mathbf{c}^{(j)})^\top) \cdot \mathbf{R}^{-1}$.
  4. Generate and return fresh handles in group 1 for $(\hat{\mathbf{u}}^{(j)})^\top$.

  Observe that $\mathsf{Expt}_1 = \mathsf{Expt}_{2,0}$, and that $\mathsf{Expt}_{2,J}$ is a generic group instantiation of $\mathsf{Expt}^{\mathsf{eFP}}_{\Pi,\mathcal{S},\mathcal{A}}(1^\lambda, \mathcal{E}_{n,t}, 1)$. This follows since the input $\mathcal{D}$ is not used by $\mathcal{O}_{\mathsf{DKeyGen}}$ at any point during the course of the experiment, so $\mathcal{O}_{\mathsf{DKeyGen}}$ can be simulated by $\mathcal{S}$.

Claim. $\mathcal{A}$ cannot distinguish between Expt_0 and Expt_1 except with $\mathsf{negl}(\lambda)$ advantage.

Proof. Let j_k denote the index input to $\mathcal{O}_{\mathsf{Enc}}(\mathsf{pk}, \cdot)$ on the kth query. We condition on the event that for each Y_k, $X_{j'} \not\subseteq Y_k$ for all $j' \neq j_k$. This occurs with overwhelming probability due to the definition of $\mathcal{E}_{n,t}$ and a union bound over $J, K = \mathsf{poly}(\lambda)$. We further condition on the event that in Expt_0, for all k, j, $(\mathbf{c}^{(j)})^\top \cdot \mathbf{w}^{(k)} = 0$ if and only if $j = j_k$, which follows from Lemma 2 and a union bound.

In both games, consider replacing all the entries of $\mathbf{R}$ and all $\alpha_j, \beta_k, \gamma_k$ with formal variables, and call the resulting games $\mathsf{Sim\text{-}Real}'$ and $\mathsf{Sim\text{-}Enc}'$. By a similar argument as in the proof of Theorem 5.1, $\mathcal{A}$ notices this switch with negligible probability. Now fix any zero-test query that $\mathcal{A}$ submits. We claim that it evaluates to identically zero in $\mathsf{Sim\text{-}Real}'$ if and only if it does so in $\mathsf{Sim\text{-}Enc}'$.

As in the proof of Theorem 5.1, we first write explicitly the form of a zero-test query in $\mathsf{Sim\text{-}Real}'/\mathsf{Sim\text{-}Enc}'$. Define $(\hat{\mathbf{u}}'^{(j)})^\top := (\hat{\alpha}_j^{-1} \mid (\mathbf{c}^{(j)})^\top) \cdot \hat{\mathbf{R}}^{-1}$, and $\hat{\mathbf{v}}'^{(k)} := \hat{\mathbf{R}} \cdot (\hat{\beta}_k^{-1}\hat{\gamma}_k \mid (\mathbf{w}^{(k)})^\top)^\top$. Letting $\kappa_{j,m}, \tau_{k,\ell}, \delta_{k,\ell}, \eta_{j,m,k,\ell}, \rho_{j,m,k,\ell}, \nu_k$ refer to the coefficients submitted by $\mathcal{A}$, the general form of a zero-test query is

$$\sum_{j,m} \kappa_{j,m}\hat{\alpha}_j \hat{\mathbf{u}}'^{(j)}_m + \sum_{k,\ell} \tau_{k,\ell}\hat{\mathbf{v}}'^{(k)}_\ell \hat{\beta}_k + \sum_{k,\ell} \delta_{k,\ell}\hat{r}_{k,\ell} + \sum_{j,m,k,\ell} \eta_{j,m,k,\ell}\hat{\alpha}_j (\hat{\mathbf{u}}'^{(j)}_m)^\top \hat{\mathbf{v}}'^{(k)}_\ell \hat{\beta}_k + \sum_{j,m,k,\ell} \rho_{j,m,k,\ell}\hat{\alpha}_j \hat{\mathbf{u}}'^{(j)}_m \hat{r}_{k,\ell} + \sum_k \nu_k(\mu'_k + \hat{\gamma}_k).$$

First, notice that all but the second and fourth terms are identical between the two games $\mathsf{Sim\text{-}Real}'$ and $\mathsf{Sim\text{-}Enc}'$, since the only difference lies in the $\mathbf{v}'^{(k)}$ vectors. Note further that an adversary can only hope to obtain a successful zero-test in either game by setting $\tau_{k,\ell} = 0$ for all k, ℓ. This follows from a similar argument as in the proof of Theorem 5.1, where the entire expression is stratified by the $\hat{\beta}_k$ variables. Looking at each $\hat{\beta}_k$ term, it is clear that the formal variables from $\hat{\mathbf{R}}$ in the elements of the $\hat{\mathbf{v}}'^{(k)}$ vectors cannot be canceled out.

Thus we focus on the fourth term, and stratify by the $\hat{\alpha}_j$ and $\hat{\beta}_k$ variables to obtain

$$\sum_{j,k} \hat{\alpha}_j \hat{\beta}_k \left(\sum_{m,\ell} \eta_{j,m,k,\ell} (\hat{\mathbf{u}}'^{(j)}_m)^\top \hat{\mathbf{v}}'^{(k)}_\ell \right).$$

$\mathcal{A}$ can only hope to obtain a successful zero-test if the coefficient on each $\hat{\alpha}_j\hat{\beta}_k$ is a constant multiple of $\hat{\alpha}_j^{-1}\hat{\beta}_k^{-1}$. So by Lemma 3, for this to happen, it must be the case that for each (j, k), the coefficients $\{\eta_{j,m,k,\ell}\}_{m,\ell}$ induce a scaling of the inner product between $\hat{\mathbf{u}}'^{(j)}$ and $\hat{\mathbf{v}}'^{(k)}$. For each (j, k), let $z_{j,k}$ be this scaling. Now we can re-write this term as

$$\sum_{j,k} \hat{\alpha}_j \hat{\beta}_k z_{j,k} \left(\hat{\alpha}_j^{-1} \hat{\beta}_k^{-1} \hat{\gamma}_k + (\mathbf{c}^{(j)})^\top \cdot \mathbf{w}^{(k)} \right) =$$

$$\sum_k \hat{\gamma}_k \left(\sum_j z_{j,k} \right) + \sum_{j,k} \hat{\alpha}_j \hat{\beta}_k z_{j,k} (\mathbf{c}^{(j)})^\top \cdot \mathbf{w}^{(k)}.$$

Again notice that the first term will be identical in both games, so focus attention on the second. We see that the term will be zero if and only, for each (j, k) such that $z_{j,k} \neq 0$, $(\mathbf{c}^{(j)})^\top \cdot \mathbf{w}^{(k)} = 0$. Finally, we see that $(\mathbf{c}^{(j)})^\top \cdot \mathbf{w}^{(k)} = 0$ under the exact same conditions in both games, namely, if and only $j = j_k$ (due to the conditioning at the beginning of this proof). This completes the proof of the claim. $\square$

Claim. For $\ell = 1, \ldots, J$, $\mathcal{A}$ cannot distinguish between $\mathsf{Expt}_{2,\ell-1}$ and $\mathsf{Expt}_{2,\ell}$ except with $\mathsf{negl}(\lambda)$ advantage.

Proof. This follows from a straightforward reduction to the generic group security of small superset obfuscation with the simulator specified in the proof of Theorem 1 (which initializes the adversary with $t+1$ uniformly random group elements). Let $\hat{W}$ refer to the set of formal variables $\{\hat{w}_{k,i}\}_{k\in[K],i\in[t]}$. Notice that the only difference between $\mathsf{Expt}_{2,\ell-1}$ and $\mathsf{Expt}_{2,\ell}$ is whether $\mathbf{c}^{(\ell)}$ is a uniformly random vector or an obfuscation of X_ℓ. Consider a reduction $\mathcal{B}$ interacting with the generic group model game for small superset obfuscation. $\mathcal{B}$ associates the $t+1$ handles it receives with $\mathbf{c}^{(\ell)}$, which it sets to be formal variables $\hat{c}_1, \ldots, \hat{c}_{t+1}$. Let $\hat{C}$ refer to this set of formal variables. It can now simulate $\mathsf{Expt}_{2,\ell-1}$ or $\mathsf{Expt}_{2,\ell}$ for $\mathcal{A}$, maintaining its table with polynomials over $\hat{C}$ and $\hat{W}$. Whenever $\mathcal{A}$ make a zero-test query, $\mathcal{B}$ stratifies the resulting polynomial by the $\hat{W}$ variables, considering separately each coefficient on $\hat{w}_{k,i}$. Note that by the restrictions imposed by the bilinear generic group model, each such coefficient must be a linear polynomial over the $\hat{C}$ variables. Therefore, $\mathcal{B}$ can determine whether it is zero via a zero-test query to its own generic group oracle. Combining the results, $\mathcal{B}$ can respond appropriately to $\mathcal{A}$. If $\mathcal{B}$'s generic group oracle is implementing the valid obfuscation, then $\mathcal{A}$ sees exactly $\mathsf{Expt}_{2,\ell-1}$. If $\mathcal{B}$'s generic group oracle is initializing $\mathcal{B}$ with $t+1$ random elements, then $\mathcal{A}$ sees exactly $\mathsf{Expt}_{2,\ell}$. This completes the proof of the claim. $\square$

Acknowledgements

This research was supported in part by ARO and DARPA Safeware under contracts W911NF-15-C-0227, W911NF-15-C-0236, W911NF-16-1-0389, W911NF-15-C-0213, and by NSF grants CNS-1633282, 1562888, 1565208, and 1814919. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARO and DARPA.

References

AAB$^+$15. Shashank Agrawal, Shweta Agrawal, Saikrishna Badrinarayanan, Abishek Kumarasubramanian, Manoj Prabhakaran, and Amit Sahai. On the practical security of inner product functional encryption. In *PKC 2015*, 2015.

ABF16. Afonso Arriaga, Manuel Barbosa, and Pooya Farshim. Private functional encryption: Indistinguishability-based definitions and constructions from obfuscation. In *INDOCRYPT 2016*, 2016.

BBC$^+$14. Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *TCC 2014*, 2014.

BCKP14. Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In *CRYPTO 2014, Part II*, 2014.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, 2001.

BGI$^+$01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, 2001.

BGMZ18. James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In *TCC 2018, Part II*, 2018.

BKM$^+$18. Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. A simple obfuscation scheme for pattern-matching with wildcards. In *CRYPTO 2018, Part III*, 2018.

BLMZ19. James Bartusek, Tancrède Lepoint, Fermi Ma, and Mark Zhandry. New techniques for obfuscating conjunctions. In *EUROCRYPT 2019, Part III*, 2019.

BMSZ16. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *EUROCRYPT 2016, Part II*, 2016.

BR13. Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In *CRYPTO 2013, Part II*, 2013.

BR17. Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. *Journal of Cryptology*, (1), 2017.

BRS13a. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO 2013, Part II*, 2013.

BRS13b. Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private subspace-membership encryption and its applications. In *ASIACRYPT 2013, Part I*, 2013.

BSW09. John Bethencourt, Dawn Song, and Brent Waters. New techniques for private stream searching. *ACM Transactions on Information and System Security (TISSEC)*, 12(3):16, 2009.

BW07. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, 2007.

BW19. Ward Beullens and Hoeteck Wee. Obfuscating simple functionalities from knowledge assumptions. In *PKC 2019, Part II*, 2019.

Can97. Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO'97*, 1997.

CRV10. Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC 2010*, 2010.

DS05. Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In *37th ACM STOC*, 2005.

GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th FOCS*, 2017.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *43rd ACM STOC*, 2011.

ITZ16. Vincenzo Iovino, Qiang Tang, and Karol Zebrowski. On the power of public-key function-private functional encryption. In *CANS 16*, 2016.

Jou04. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, (4), 2004.

KLM$^+$18. Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In *SCN 18*, 2018.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, 2008.

KSW13. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *Journal of Cryptology*, (2), 2013.

LPS04. Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT 2004*, 2004.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In *10th IMA International Conference on Cryptography and Coding*, 2005.

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In *CRYPTO 2003*, 2003.

Nec94. V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

OS07. Rafail Ostrovsky and William E. Skeith. Private searching on streaming data. *Journal of Cryptology*, (4), 2007.

PM18. Sikhar Patranabis and Debdeep Mukhopadhyay. New lower bounds on predicate entropy for function private public-key predicate encryption. Cryptology ePrint Archive, Report 2018/190, 2018. https://eprint.iacr.org/2018/190.

PMR19. Sikhar Patranabis, Debdeep Mukhopadhyay, and Somindu C. Ramanna. Function private predicate encryption for low min-entropy predicates. In *PKC 2019, Part II*, 2019.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT'97*, 1997.

SSW09. Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC 2009*, 2009.

SWP00. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, 2000.

Wee05. Hoeteck Wee. On obfuscating point functions. In *37th ACM STOC*, 2005.

WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th FOCS*, 2017.