Bi-Deniable Public-Key Encryption

Adam O'Neill[1,⋆], Chris Peikert[2,⋆⋆], and Brent Waters[1,⋆⋆⋆]

[1] University of Texas at Austin
[2] Georgia Institute of Technology

Abstract. In 1997, Canetti *et al.* (CRYPTO 1997) put forward the intruiging notion of *deniable encryption*, which (informally) allows a sender and/or receiver, having already performed some encrypted communication, to produce 'fake' (but legitimate-looking) random coins that open the ciphertext to another message. Deniability is a powerful notion for both practice and theory: apart from its inherent utility for resisting coercion, a deniable scheme is also noncommitting (a useful property in constructing adaptively secure protocols) and secure under selective-opening attacks on whichever parties can equivocate. To date, however, known constructions have achieved only limited forms of deniability, requiring at least one party to withhold its randomness, and in some cases using an interactive protocol or external parties.

In this work we construct *bi-deniable* public-key cryptosystems, in which both the sender and receiver can simultaneously equivocate; we stress that the schemes are noninteractive and involve no third parties. One of our systems is based generically on "simulatable encryption" as defined by Damgård and Nielsen (CRYPTO 2000), while the other is lattice-based and builds upon the results of Gentry, Peikert and Vaikuntanathan (STOC 2008) with techniques that may be of independent interest. Both schemes work in the so-called "multi-distributional" model, in which the parties run alternative key-generation and encryption algorithms for equivocable communication, but claim under coercion to have run the prescribed algorithms. Although multi-distributional deniability has not attracted much attention, we argue that it is meaningful and useful because it provides credible coercion resistance in certain settings, and suffices for all of the related properties mentioned above.

[⋆] Supported in part by the grants of the third author. Most of this work was completed while at the Georgia Institute of Technology.

[⋆⋆] This material is based upon work supported by the National Science Foundation under Grant CNS-0716786 and CAREER Award CCF-1054495, and by the Alfred P. Sloan Foundation. 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 National Science Foundation.

[⋆⋆⋆] Department of Computer Science, University of Texas at Austin. Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA PROCEED, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Foundation, and Microsoft Faculty Fellowship.

1 Introduction

Suppose that Eve has two children: Alice, who is away at college, and a young Bob, who still lives at home. The siblings are planning a surprise party for Eve, so to keep their plans secret, they communicate using public-key encryption. Eve, however, has taken note of their encrypted communications and grows suspicious. Using her inherent parental authority, she demands that Alice and Bob reveal their secret decryption keys, as well as any of the encryption randomness they might have retained. Is there any way for Alice and Bob to comply, without spoiling the surprise? The answer seems to be obviously no: using the secret keys, Eve can simply decrypt their messages and learn about the party.

However, the above argument misses a subtle point: if Alice and Bob are able to produce *alternative* keys and randomness that are consistent with their ciphertexts so far, then they might be able to fool Eve into thinking that they are communicating about something else (or at least not alert her to the party). A scheme that makes this possible is said to be *deniable*, a notion formally introduced by Canetti, Dwork, Naor, and Ostrovsky [10].

In practice, deniable encryption has been sought by users whose legitimate activities may not always be protected from subpoenas or legal coercion, e.g., journalists and whistleblowers, or lawyers and activists in repressive regimes. Indeed, several commercial and open-source storage encryption products claim limited forms of deniability (see, for example, [1, 2], and further references in [24]), though without formal definitions or supporting security proofs. More worryingly, these products only allow for denying the *existence* of messages on a storage medium, not for *equivocating* those messages. This is insufficient in a communications setting, where the mere exchange of messages between parties indicates that they are communicating in some form.

Deniability is also a compelling property for theoretical reasons: in particular, deniable encryption schemes are *noncommitting* (a fundamental concept in the design of adaptively secure protocols) [11, 16, 14], secure against *selective-opening* attacks [19, 7], and imply incoercible multiparty computation [12]. We point out that deniable encryption is stronger than noncommitting encryption, because equivocable ciphertexts actually decrypt to the intended messages, and users of the system (not just a simulator) can themselves produce such ciphertexts.

Canetti *et al.* distinguish between two different models of deniability. The first is *full* deniability, in which the parties always run the prescribed key-generation and encryption algorithms, and can equivocate their messages later on if they so choose. The second model is called *multi-distributional* deniability, in which there exist alternative "deniable" algorithms whose outputs can be equivocated, so that it appears as if the *prescribed* algorithms had been used all along. Whether these models are useful in various settings has been the subject of some debate over the years; we discuss these issues in Section 1.2 below. We also discuss some recent developments and related work in Section 1.3.

Under standard assumptions, Canetti *et al.* construct a multi-distributional *sender*-deniable scheme (i.e., one that remains secure if only the sender is coerced), and give a *fully* sender-deniable scheme where the coercer's distinguishing advantage between a 'real' and 'fake' opening is an inverse polynomial that depends on the public key size. They also construct a receiver-deniable scheme that requires an additional round of interaction, and a sender- and receiver-deniable protocol that relies on third parties, at least one of whom must remain uncoerced. In particular, up to this point there have not been any noninteractive schemes offering receiver-deniability, nor any schemes (interactive or not) in which all the parties can be coerced simultaneously, in either of the two models (full or multi-distributional deniability).

1.1 Our Contributions

Bideniable public-key encryption. Our main results are the first known *bideniable* (that is, simultaneously sender- and receiver-deniable) public-key encryption schemes, in the multi-distributional model. We stress that the schemes are noninteractive, require no third parties, and are immediately noncommitting and secure under selective-opening attacks.

We give two qualitatively different constructions. The first is built generically, using a combinatorial construction with somewhat large overhead, from any "simulatable" encryption scheme in the sense of Damgård and Nielsen [16]. This shows (perhaps surprisingly) that simulatability is sufficient not only for noncommitting encryption, but for a form of deniability as well. The scheme is presented in Section 4.

Our second scheme is based on worst-case lattice problems via "learning with errors" [23], and builds upon the trapdoor function and identity-based encryption techniques of Gentry, Peikert, and Vaikuntanathan [21]. In particular, it exploits a unique property of the GPV IBE, namely its negligible chance of oblivious decryption error when the secret key vector and the error vector in the ciphertext are too highly "aligned." Our scheme relies on the ability of the receiver to resample a fresh secret key that is highly correlated with the ciphertext error term. Proving that this secret key "looks real" relies on a symmetry between correlated (discrete) Gaussians, which we believe may be of independent interest and application in lattice cryptography. Interestingly, the deniable scheme is essentially identical to the GPV cryptosystem, and it appears to be the first known cryptosystem that "naturally" supports receiver-deniability without any substantial changes. It is also essentially as efficient for the receiver as the GPV system, but it is less efficient for the sender because she must encrypt each bit separately (rather than amortizing). The details of the system are described in Section 6.

In addition to our public-key schemes, we also define notions of deniability for the identity-based setting, and show how our techniques immediately adapt to it as well. As we discuss below, multi-distributional deniability (especially for the receiver) may be more palatable in this setting because the receiver does not run a different key-generation algorithm (indeed, there is no such algorithm). We also remark that to be meaningful, the identity-based setting inherently requires any solution to be noninteractive.

Plan-ahead bideniability with short keys. A simple information-theoretic argument reveals that in any noninteractive receiver-deniable encryption scheme, the secret key must be at least as long as the message: a fixed ciphertext can only decrypt to N different plaintexts if there are at least N distinct secret keys for the public key (see also [22] and the recent work [9]). We circumvent this constraint by designing a scheme offering *"plan-ahead"* bideniability (a notion also introduced in [10]) that can encrypt arbitrarily long messages using *fixed-sized* keys. In plan-ahead deniability, the sender must choose at encryption time a bounded number of potential 'fake' messages, to which the parties may later equivocate. This may be seen as the deniable analogue of "somewhat noncommitting encryption," introduced by Garay, Wichs and Zhou [20]. In many cases, this model would seem to be sufficient for coercion resistance, since the sender can just include one "innocuous" message along with the real one.

Our plan-ahead scheme is a hybrid system that reduces the deniable encryption of an arbitrary-length message to that of a short symmetric key. For example, when combined with the moderately good efficiency of our GPV-style bideniable scheme, the overall system is potentially usable in practice. (Though as noted above, our scheme is not able to amortize many bits into one ciphertext as the GPV scheme does, so our bandwidth requirements are larger.) Due to space constraints, we defer the details of our plan-ahead scheme to the full version.

Relations among notions. We clarify and study relations among the various types of deniability introduced in [10]. Our main contribution is that any type of *multi-distributional* deniability suffices to obtain the corresponding type of *full* deniability, with an inverse-polynomial distinguishing advantage related to the size of the public key. This further reinforces the usefulness of multi-distributional deniability itself. We also observe that for multi-distributional schemes, bideniability implies sender-deniability, but perhaps surprisingly, it may not imply receiver-deniability alone. That is, bideniability relies on the sender to correctly run the deniable encryption algorithm.

1.2 Discussion

Is (multi-distributional) deniability useful? The ideal deniable encryption scheme would be noninteractive and fully deniable for both parties. Unfortunately, it has been recently shown [9] that these properties cannot all be achieved at once, even for receiver deniability alone (see related work below). So to obtain a noninteractive scheme we must work in the multi-distributional model.

A common objection to multi-distributional deniability is that, since there are alternative deniable algorithms (for encryption and key generation) that are strictly more powerful than the normal ones, why would anyone ever run the normal algorithms? And given this situation, why would a coercer ever accept a transcript corresponding to the normal algorithms? Whether this is a significant problem will depend on the setting in which coercion happens, and what recourse the coercer has in response to the users' claims.

For example, if there is a prescribed legal process (such as a subpoena or search warrant) by which parties are forced to reveal their transcripts, then multi-distributional deniability may be sufficient to protect the users. Even if the coercer asks for a transcript corresponding to the deniable algorithms, the users can simply assert that they did not run those algorithms, and so cannot produce coins for them. The users' claims might also gain in credibility via "safety in numbers," if a deployed implementation defaults to the normal algorithms — which do make up an operational cryptosystem, after all — or by formally standardizing on the normal algorithms within an organization. Since the coercer would only have *reason to believe* — but not any actual *evidence* — that the deniable algorithms were used in a particular instance, imposing a sanction seems fundamentally unjust, and might even be ruled out by the prescribed process. If, on the other hand, the coercer is able to punish the users until they "tell him what he wants to hear," then multi-distributional deniability might not be enough to protect the users — but neither might full deniability! After all, in either model the coercer has no reason to believe what the users have revealed. Anticipating potential punishment, the users of a multi-distributional scheme might retain the coins of the deniable algorithms as a "backup plan," just in case they might later want to reveal a convincing proof for the true message (e.g., if the punishment becomes too severe). But even a fully deniable scheme allows for a similar backup plan and proof of the true message, by using "verifiably random" coins such as the digits of π or the output of a pseudorandom generator.

In the identity-based setting, multi-distributional deniability may be useful as well, especially for the receiver. Here the receiver does not run a key-generation algorithm at all, but instead gets his secret key from an authority who possesses a 'master' secret key for all users. Our model allows the receiver to ask the authority for a fake (but real-looking) secret key that causes a particular ciphertext to decrypt to any desired message. This could be useful if the authority is out of the coercer's jurisdiction (e.g., if the receiver is travelling in another country), or if it can argue that exposing its master secret key would harm the privacy of too many other users.

In summary, the purpose of deniability is not at all to 'convince' the coercer that the surrendered transcripts are real; indeed, it is common knowledge that they can easily be faked. Instead, the goal is to *preempt coercion* in the first place by making it useless, since parties who "stick to their stories" can never be pinned down to the real message. At the same time, neither form of deniability seems appropriate if a user might eventually want to convincingly reveal the true plaintext, e.g., to sell her vote in an election. The main significant difference we see between the two models relates not to security, but usability: multi-distributional deniability requires the users to know in advance which messages they might want to equivocate, whereas full deniability allows the user to decide afterward.

Why not erase? At first glance, erasures appear to provide a very simple way of achieving deniability: the parties can just tell the coercer that they deliberately erased their coins (perhaps according to a published schedule), and therefore cannot surrender them. For sender deniability, this claim might be credible, since there is no point in the sender keeping her ephemeral encryption coins. (And indeed, none of our results preclude using erasures on the sender side.) For receiver deniability, erasures seem much less credible, since the receiver must store some form of decryption key in order to decrypt messages, and at some point in time this key can be subject to coercion. Certain regulatory regimes (e.g., in the financial sector) might also mandate retention of all data for a certain time, for potential audit. The existence of deniable encryption means that such requirements would still not necessarily guarantee compliance with the intent of the regulations. In any case, we contend that there is a significant qualitative difference between deniable schemes that use erasures, and those that do not. In the former, the coerced parties must resort to the claim that they no longer have the desired evidence, even though they once did. In the latter, the parties can credibly claim to have provided the coercer with all the evidence they have ever had, and yet still equivocate.

1.3 Other Related Work

Subsequent to the initial dissemination of this work in 2010, there has been additional work on deniable encryption that complements our own. Dürmuth and Freeman [18] announced an interactive, fully sender-deniable encryption protocol (i.e., one with a single encryption protocol and negligible detection advantage). However, following its publication Peikert and Waters found a complete break of this system (and a corresponding flaw in the claimed security proof); see [17] for details. In particular, the problem of constructing a fully deniable encryption scheme remains an intriguing open question. Bendlin *et al.* [9] recently showed that any noninteractive public-key scheme having key size σ can be fully receiver-deniable (or bideniable) only with non-negligible $\Omega(1/\sigma)$ detection advantage. In particular, our use of the multi-distributional model is necessary to achieve a noninteractive receiver-deniable scheme.

Deniability can be seen as a type of security that holds true even when secret information is revealed to the adversary. In the case of break-ins, one relevant notion is "forward security" (see, e.g., [8, 13]), which relies on secure erasures to update secret state over time. In the case of side-channel or memory attacks, relevant notions include the "bounded retrieval model" and "leakage-resilient" cryptography, which limit the amount of secret information the adversary may learn (see the recent survey [5] and references therein). In contrast, deniability ensures security in contexts where the adversary obtains the *entire, unchanging* secret key and/or encryption randomness, but cannot tell whether those values came from the actual execution of the system.

2 Preliminaries

We denote the set of all binary strings by $\{0,1\}^*$ and the set of all binary strings of length i by $\{0,1\}^i$. The length of a string s is denoted by $|s|$. By $s_1 \| s_2$ we denote an encoding of strings s_1, s_2 from which the two strings are unambiguously recoverable. (If the lengths of s_1, s_2 are known, then concatenation suffices.) For $i \in \mathbb{N}$ we denote by $[i]$ the set $\{1, \ldots, i\}$. We use the standard definitions of negligible functions, and statistical and computational indistinguishability.

We say an algorithm A with input space $\mathcal{X}$ has *invertible sampling* [16] if there is an efficient inverting algorithm, denoted I_A, such that for all $x \in \mathcal{X}$ the outputs of the following two experiments are indistinguishable (either computationally, or statistically):

$$\begin{array}{l|l} y \leftarrow A(x;r) & y \leftarrow A(x;r) \\ & r' \leftarrow I_A(x,y) \\ \text{Return } (x,y,r) & \text{Return } (x,y,r') \end{array}$$

In other words, given just an input-output pair of A, it is possible to efficiently generate appropriately distributed randomness that "explains" it. It may also be the case that I_A requires some "trapdoor" information about x in order to do so. Namely, we say that A has *trapdoor* invertible sampling if we replace the second line in the right-hand experiment above with "$r' \leftarrow I_A(td_x, x, y)$," where td_x is a trapdoor corresponding to x (we think of x and td_x as being sampled jointly as the setup to both of the above experiments).

A *public-key cryptosystem* PKC with message space $\mathcal{M}$ consists of three algorithms: The *key generation algorithm* $\mathsf{Gen}(1^n; r_R)$ outputs a public key pk, and the randomness r_R is used as the associated secret decryption key. (This convention is natural in the context of deniability, where we might even consider coercing the receiver to reveal the random coins r_R it used to generate its public key. This is without loss of generality, since the stored "secret key," whatever its form, can always be computed from r_R.) The *encryption algorithm* $\mathsf{Enc}(pk, m; r_S)$ outputs a ciphertext c. The deterministic *decryption algorithm* $\mathsf{Dec}(pk, r_R, c)$ outputs a message m or $\perp$. For correctness, we require the probability that $m' \neq m$ be negligible for all messages $m \in \mathcal{M}$, over the experiment $pk \leftarrow \mathsf{Gen}(1^n; r_R)$, $c \leftarrow \mathsf{Enc}(pk, m)$, $m' \leftarrow \mathsf{Dec}(pk, r_R, c)$. To distinguish between the above notion and deniable encryption as defined in Section 3, we sometimes refer to the former as *normal* encryption.

3 Bideniable Encryption

Here we formally define bideniable encryption and its security properties, along with some weaker variants. (We use "bideniable" as shorthand for "sender-and-receiver deniable" in the language of Canetti *et al.* [10].) Following the definitions, we discuss further how our notion relates to the variants of deniable encryption introduced in [10].

As with normal encryption, bideniable encryption allows a sender in possession of the receiver's public key to communicate a message to the latter, confidentially. Additionally, if the parties are later coerced to reveal all their secret data — namely, the coins used by the sender to encrypt her message and/or those used by the receiver to generate her key — bideniable encryption allows them to do so as if *any desired message* (possibly chosen as late as at the time of coercion) had been encrypted.

In a *multi-distributional* deniable encryption scheme, there are 'normal' key generation, encryption, and decryption algorithms that can be run as usual

— though the resulting communication may not be equivocable later on. In addition, there are 'deniable' key generation and encryption algorithms that can be used for equivocable communication. Associated with these deniable algorithms are 'faking' algorithms, which can generate secret coins that open a deniably generated public key and ciphertext to any desired message, as if the *normal* algorithms had been used to generate them. Note that the ability to generate fake *random coins* for the parties yields the strongest definition, since such coins can be used to compute whatever locally stored values (e.g., a secret key of some form) the coercer might expect. We now give a formal definition.

Definition 1 (Deniable encryption). *A* multi-distributional sender-*,* receiver-*, or* bi-deniable *encryption scheme DEN with message space $\mathcal{M}$ is made up of the following algorithms:*

- *The* normal *key-generation, encryption, and decryption algorithms* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are defined as usual for public-key encryption (see Section 2). These algorithms make up the* induced normal scheme*.*
- *The* deniable *key-generation algorithm* $\mathsf{DenGen}(1^n)$ *outputs* (pk, fk)*, where* fk *is the* faking key.[1] *We also extend* Dec *to so that it can decrypt using* fk *in lieu of the usual receiver randomness* r_R*.*
- *The* deniable *encryption algorithm* DenEnc *has the same interface as the normal encryption algorithm.*
- *The* sender faking algorithm $\mathsf{SendFake}(pk, r_S, m', m)$*, given a public key* pk*, original coins* r_S *and message* m' *of* DenEnc*, and desired message* m*, outputs faked random coins* r_S^* *for* Enc*.*
- *The* receiver faking algorithm $\mathsf{RecFake}(pk, fk, c, m)$*, given the public and faking keys* pk *and* fk *(respectively), a ciphertext* c*, and a desired message* m*, outputs faked random coins* r_R^* *for* Gen*.*

We require the following properties:

1. Correctness. *Any triplet* $(\mathsf{G}, \mathsf{E}, \mathsf{Dec})$*, where* $\mathsf{G} \in \{\mathsf{Gen}, \mathsf{DenGen}\}$ *and* $\mathsf{E} \in \{\mathsf{Enc}, \mathsf{DenEnc}\}$*, should form a correct public-key encryption scheme.*
2. Multi-distributional deniability. *Let* $m, m' \in \mathcal{M}$ *be arbitrary messages, not necessarily different. The appropriate experiment below, which represents equivocation (by the appropriate party/parties) of an encrypted* m' *as* m*, should be computationally indistinguishable from the following 'honest opening' experiment: let* $pk \leftarrow \mathsf{Gen}(1^n; r_R)$*,* $c \leftarrow \mathsf{Enc}(pk, m; r_S)$*, and output* pk*,* c*, and whichever of* r_R*,* r_S *are appropriate to the type of deniability under consideration.*

[1] Without loss of generality, we could replace fk with the randomness of DenGen, but since this randomness will never be exposed to the adversary, we elect to define a distinguished faking key.

Sender-Deniable	*Receiver-Deniable*	*Bi-Deniable*
$pk \leftarrow \mathsf{Gen}(1^n; r_R)$	$(pk, fk) \leftarrow \mathsf{DenGen}(1^n)$	$(pk, fk) \leftarrow \mathsf{DenGen}(1^n)$
$c \leftarrow \mathsf{DenEnc}(pk, m'; r_S)$	$c \leftarrow \mathsf{Enc}(pk, m'; r_S)$	$c \leftarrow \mathsf{DenEnc}(pk, m'; r_S)$
	$r_R^* \leftarrow \mathsf{RecFake}(pk, fk, c, m)$	$r_R^* \leftarrow \mathsf{RecFake}(pk, fk, c, b)$
$r_S^* \leftarrow \mathsf{SendFake}(pk, r_S, m', m)$		$r_S^* \leftarrow \mathsf{SendFake}(pk, r_S, m', m)$
Return (pk, c, r_S^*)	*Return* (pk, c, r_R^*)	*Return* (pk, c, r_R^*, r_S^*)

Multi-distributional bideniability is a particularly strong theoretical notion. For example, it immediately implies non-committing encryption as defined in [11] — but in addition, equivocable ciphertexts actually decrypt to the intended messages, and can be produced by the regular users of the scheme, not just by a simulator. Bideniability is also important in practice; in particular, each party's security does not depend upon the incoercibility of the other.

Note that we did not explicitly require DEN to satisfy the standard notion of indistinguishability under chosen-plaintext attack; this is because it is implied by any of the above notions of deniability.

Proposition 1. *Suppose that DEN satisfies any of sender-, receiver-, or bideniability. Then any triplet* $(\mathsf{G}, \mathsf{E}, \mathsf{Dec})$*, where* $\mathsf{G} \in \{\mathsf{Gen}, \mathsf{DenGen}\}$ *and* $\mathsf{E} \in \{\mathsf{Enc}, \mathsf{DenEnc}\}$*, is semantically secure.*

While our focus is on multi-distributional bideniability, we also briefly examine interrelations among the other types. We start with a basic question: for a particular deniable encryption scheme DEN, which notions of deniability imply which others? (This question is also important in practice, since an encryption scheme may not always be used in the ways it is intended.) First, we show that bideniablility implies sender deniability.

Proposition 2. *If DEN is bideniable, then DEN is sender-deniable.*

Proof. By assumption, we know that the distributions (pk, c, r_R, r_S) and (pk, c, r_R^*, r_S^*) are computationally indistinguishable, as produced by the two bideniability experiments. Clearly, the distributions (pk, c, r_S) and (pk, c, r_S^*) are indistinguishable when produced by the same two experiments, where we can now omit the generation of r_R^* in the faking experiment. This modified bideniable faking experiment producing (pk, c, r_S^*) differs from the sender-deniable faking experiment only its use of DenGen instead of Gen. Because neither experiment uses fk, all we need is for the pks output by DenGen and by Gen to be indistinguishable. But this follows directly from bideniability, by restricting the outputs of the two experiments to just pk.

One might also expect bideniability to imply receiver-deniability. Perhaps surprisingly, at least in the multi-distributional setting this appears not to be the case! For example, in our abstract scheme from Section 5, the receiver can equivocate a *normal* ciphertext in one direction (from 1 to 0), but apparently not from 0 to 1. In general, the problem is that to allow the receiver to equivocate a message, DenEnc may need to produce special ciphertexts that Enc would never (or very rarely) output. In other words, the receiver's ability to equivocate a message may depend crucially the sender's use of DenEnc.

In the reverse direction, it appears that a scheme that is both sender-deniable and (separately) receiver-deniable still might not be bideniable. Indeed, the fake randomness produced by $\mathsf{SendFake}$ and $\mathsf{RecFake}$ (which depend on the ciphertext c) might be obviously correlated in such a way that exposing both together is easily detected, whereas exposing only one is safe. Constructing a concrete example along these lines to demonstrate a formal separation remains an interesting problem.

4 Bideniable Encryption from Simulatable Encryption

Here we give a bideniable public-key encryption scheme from any *simulatable* one, in the sense of Damgård and Nielsen [16]. In particular, this shows that simulatable encryption suffices to realize not just a noncommitting encryption scheme, but also a (multi-distributional) bideniable one.

Simulatable public-key encryption. We recall the notion of a simulatable public-key encryption scheme from [16]. Intuitively, this is a scheme in which it is possible to 'obliviously' sample a public key without knowing the secret key, and to 'obliviously' sample the encryption of a random message without knowing the message.

Definition 2 (Simulatable PKE [16]). *A* simulatable public-key encryption scheme *PKC-SIM with message space $\mathcal{M}$ is made up of the following algorithms:*

- *The* normal *key-generation, encryption, and decryption algorithms* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *are defined as usual for a public-key encryption scheme (see Section 2).*
- *The* oblivious *key-generation algorithm* $\mathsf{OGen}(1^n; r_{\mathsf{OGen}})$ *outputs a public key opk, and has invertible sampling via algorithm* I_{OGen}.
- *The* oblivious *encryption algorithm* $\mathsf{OEnc}(pk)$ *outputs a ciphertext oc, and has invertible sampling via algorithm* I_{OEnc}.

We require the following properties:

1. Oblivious key generation. *The distribution of pk, where* $pk \leftarrow \mathsf{Gen}(1^n; r_R)$, *should be computationally indistinguishable from opk, where* $opk \leftarrow \mathsf{OGen}(1^n; r_{\mathsf{OGen}})$.
2. Oblivious ciphertext generation. *The output distributions of the following two experiments should be computationally indistinguishable:*

$$\begin{array}{l|l} pk \leftarrow \mathsf{Gen}(1^n; r_R) & pk \leftarrow \mathsf{Gen}(1^n; r_R) \\ m \leftarrow \mathcal{M} & \\ c \leftarrow \mathsf{Enc}(pk, m) & oc \leftarrow \mathsf{OEnc}(pk) \\ \textit{Return } (pk, r_R, c) & \textit{Return } (pk, r_R, oc) \end{array}$$

Note that the above conditions are non-trivial in light of the the fact that OGen, OEnc *are required to have invertible sampling. In particular, it follows by Condition 2 (after removing r_R from the output distributions) that any simulatable public-key encryption scheme is semantically secure.*

Simulatable encryption can be realized under a variety of standard computational assumptions such as DDH and RSA [16], as well as worst-case lattice assumptions [16, 21] (though we later show a more efficient bideniable encryption scheme based on the latter using an entirely different approach).

4.1 A "Coordinated" Scheme

Overview. To get at the technical core of our construction, we first present a *coordinated* scheme in which the faked random coins r_R^* for the receiver and r_S^* for the sender are outputs of the *same* algorithm $\mathsf{FakeCoins}(pk, fk, r_S, b', b, c)$ (we give the corresponding ciphertext c for convenience); i.e., the sender and receiver coordinate their faked coins upon being coerced. We later describe how the coordination can be removed for our specific scheme. Additionally, we present a scheme that encrypts one-bit messages, but a scheme that encrypts poly(n)-bit messages then follows generically by parallel repetition with independent keys.

The high-level idea for our scheme draws on and extends that of Choi *et al.* [14] (who generalized the approach of Damgård and Nielsen [16]) used to achieve noncommitting encryption. In our scheme, we run $5n$ instances of an underlying simulatable encryption scheme in parallel. In normal (non-deniable) operation, by using the oblivious key and ciphertext generation algorithms appropriately, the receiver "knows" the secret keys corresponding to a random size-n subset $\mathcal{S} \subset [5n]$ of indices and the sender "knows" the plaintexts and associated encryption randomness corresponding to the ciphertexts for an independent random size-n subset $\mathcal{R} \subset [5n]$. In particular, $\mathcal{S}$ corresponds to encryptions of the message bit b, and $[5n] \setminus \mathcal{S}$ corresponds to oblivious encryptions of random bits. To decrypt, the receiver decrypts ciphertexts corresponding to $\mathcal{R}$ and takes a majority vote. In deniable operation, the receiver knows *all* the secret keys and the sender knows *all* the plaintexts and associated encryption randomness. In particular, in addition to choosing a random size-n 'biasing' subset $\mathcal{Y} \subset [5n]$ corresponding to encryptions of the true message bit b', the sender also chooses two random pairwise disjoint size-n subsets $\mathcal{S}_0, \mathcal{S}_1 \subset [5n]$, also disjoint from $\mathcal{Y}$, corresponding to encryptions of 0s and 1s respectively; the ciphertexts in $[5n] \setminus \mathcal{Y} \cup \mathcal{S}_0 \cup \mathcal{S}_1$ encrypt random bits. To open a deniably encrypted message b' as b, the sender chooses $\mathcal{S}^* = \mathcal{S}_b$ and the receiver chooses $\mathcal{R}^*$ so that it has an appropriate random number of elements in common with $\mathcal{S}_b$, and then chooses the remainder of $\mathcal{R}^*$ as random indices from $[5n] \setminus (\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{Y})$.

The scheme. To formally define our scheme, let PKC-SIM be simulatable public-key encryption scheme with message space $\{0, 1\}$. Below, for a set $\mathcal{X}$ we denote by $\mathcal{P}(\mathcal{X})$ the set of all subsets of $\mathcal{X}$ and by $\mathcal{P}_i(\mathcal{X})$ the set of all subsets of $\mathcal{X}$ of size i. Additionally, for nonnegative integers $x, y, N \geq M$, let $P_{\mathsf{HGD}}(x; N, M, y)$ denote the probability that exactly x values from $[M]$ are chosen after a total of y values are drawn uniformly at random from $[N]$, without replacement. We denote by $\mathsf{HGD}(N, M, y)$ the *hypergeometric distribution on* $[N]$ *with parameters* M, y that assigns to each $x \in \{0, \ldots, M\}$ the probability $P_{\mathsf{HGD}}(x; N, M, y)$. Below we will use parameters N, M, y polynomial in the security parameter, so we can sample efficiently from this distribution simply by running the sampling experiment. Define scheme BI-DEN[PKC-SIM] with message-space $\{0, 1\}$ as follows:

```
BI-DEN.Gen$(1^n)$:
    $\mathcal{R} \leftarrow \mathcal{P}_n([5n])$
    For $i = 1$ to $5n$ do:
    If $i \in \mathcal{R}$ then
    $pk_i \leftarrow \mathsf{Gen}(1^n; r_{R,i})$
    Else $pk_i \leftarrow \mathsf{OGen}(1^n; r_{R,i})$
    EndFor
    $pk \leftarrow pk_1 \| \ldots \| pk_{5n}$
    Return $pk$
```

```
BI-DEN.Enc$(pk, b)$:
    $\mathcal{S} \leftarrow \mathcal{P}_n([5n])$
    For $i = 1$ to $5n$ do:
    If $i \in \mathcal{S}$ then
    $c_i \leftarrow \mathsf{Enc}(pk_i, b; r_{S,i})$
    Else $c_i \leftarrow \mathsf{OEnc}(pk_i; r_{S,i})$
    EndFor
    $c \leftarrow c_1 \| \ldots \| c_{5n}$
    Return $c$
```

```
BI-DEN.Dec$((\mathcal{R}, r_R), c)$:
    For all $i \in \mathcal{R}$ do:
    $d_i \leftarrow \mathsf{Dec}(r_{R,i}, c_i)$
    EndFor
    If most $d_i$'s are 1 then
    Return 1
    Else return 0
```

```
BI-DEN.DenGen$(1^n)$:
    $\mathcal{R} \leftarrow \mathcal{P}_n([5n])$
    For $i = 1$ to $5n$ do:
    $pk_i \leftarrow \mathsf{Gen}(1^n; r_{R,i})$
    EndFor
    $pk \leftarrow pk_1 \| \ldots \| pk_{5n}$
    $r \leftarrow r_{R,1} \| \ldots \| r_{R,5n}$
    Return $(pk, (\mathcal{R}, r))$
```

```
BI-DEN.DenEnc$(pk, b')$:
    $\mathcal{S}_0 \leftarrow \mathcal{P}_n([5n])$
    $\mathcal{S}_1 \leftarrow \mathcal{P}_n([5n] \setminus \mathcal{S}_0)$
    $\mathcal{Y} \leftarrow \mathcal{P}_n([5n] \setminus (\mathcal{S}_0 \cup \mathcal{S}_1))$
    For $i = 1$ to $5n$ do:
    If $i \in \mathcal{S}_0$ then
    $c_i \leftarrow \mathsf{Enc}(pk_i, 0; r_{S,i})$
    If $i \in \mathcal{S}_1$ then
    $c_i \leftarrow \mathsf{Enc}(pk_i, 1; r_{S,i})$
    If $i \in \mathcal{Y}$ then
    $c_i \leftarrow \mathsf{Enc}(pk_i, b'; r_{S,i})$
    Else $c_i \leftarrow \mathsf{OEnc}(pk; r_{S,i})$
    EndFor
    $c \leftarrow c_1 \| \ldots \| c_{5n}$
    Return $c$
```

```
BI-DEN.FakeCoins
        $(pk, fk, r_S, b', b, c)$:
    $z \leftarrow \mathsf{HGD}(5n, n, n)$
    $\mathcal{Z} \leftarrow \mathcal{P}_z(\mathcal{S}_b)$
    $\mathcal{Z}' \leftarrow \mathcal{P}_{n-z}([5n]$
        $\setminus(\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{Y}))$
    $\mathcal{R}^* \leftarrow \mathcal{Z} \cup \mathcal{Z}'$
    $\mathcal{S}^* \leftarrow \mathcal{S}_b$
    For $i = 1$ to $5n$ do:
    If $i \in \mathcal{S}^*$ then $r^*_{S,i} \leftarrow r_{S,i}$
    Else $r^*_{S,i} \leftarrow I_{\mathsf{OEnc}}(pk_i, c_i)$
    If $i \in \mathcal{R}^*$ then $r^*_{R,i} \leftarrow r_{R,i}$
    Else $r^*_{R,i} \leftarrow I_{\mathsf{OGen}}(pk_i)$
    EndFor
    $r^*_S \leftarrow r^*_{S,1} \| \ldots \| r^*_{S,5n}$
    $r^*_R \leftarrow r^*_{R,1} \| \ldots \| r^*_{R,5n}$
    Return $(r^*_S, r^*_R)$
```

Removing the coordination. To remove the coordination between the sender and receiver in the faking algorithm, the idea is for the sender to communicate its choices of $\mathcal{S}_0, \mathcal{S}_1, \mathcal{Y}$ to the receiver *in-band* by using a separate instance of the simulatable encryption scheme. Details are deferred to the full verison.

4.2 Correctness and Security

Theorem 1. *Let PKC-SIM be a simulatable public-key encryption scheme. Then BI-DEN[PKC-SIM] is correct.*

The proof is a simple argument that relies on a Chernoff-like tail inequality for the hypergeometric distribution, and the Chernoff bound. Due to space restrictions we leave it to the full version. We now turn to security.

Theorem 2. *Let PKC-SIM be a simulatable public-key encryption scheme. Then BI-DEN[PKC-SIM] satisfies bideniability under chosen-plaintext attacks.*

We sketch the proof outline, leaving details to the full version. We consider three hybrid experiments that transition from an 'honest' opening of an 'honest' encryption of b to a 'fake' opening of a deniably encrypted bit b' as b. The first two experiments differ only in how they choose the size-n subsets $\mathcal{Y}, \mathcal{S}_0, \mathcal{S}_1, \mathcal{R} \subset [5n]$ of indices (i.e., the corresponding key pairs and ciphertexts are generated identically; in fact, $\mathcal{S}_{1-b}, \mathcal{Y}$ are not used). Namely, the first experiment chooses $\mathcal{S}_b$ and $\mathcal{R}$ at random independently and then S_{1-b} and $\mathcal{Y}$ at random as pairwise disjoint and disjoint from $\mathcal{S}_b \cup \mathcal{R}$, while the second chooses S_b and S_{1-b} at random as pairwise disjoint, then $\mathcal{Y}$ at random as disjoint from $\mathcal{S}_b \cup \mathcal{S}_{1-b}$, and finally $\mathcal{R}$ to have an HGD-distributed random number of elements in common with $\mathcal{S}_b$ and the remainder at random disjoint from $\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{Y}$. The outputs of these experiments are identically distributed by a combinatorial argument. Finally, the third experiment appropriately changes how the key pairs and ciphertexts corresponding to $\mathcal{Y}, \mathcal{S}_0, \mathcal{S}_1, \mathcal{R}$ are generated. The outputs of the last two experiments are computationally indistinguishable by simulatability of the underlying encryption scheme.

5 Bideniable Encryption from Bitranslucent Sets

Here we construct a bideniable encryption scheme based on a new primitive we call a *bitranslucent set*, which extends the notion of a translucent set from [10]. Whereas translucent sets can be constructed straightforwardly from any trapdoor permutation (and other specific assumptions) [10], bitranslucent sets appear much more challenging to realize. In Section 6 we give a novel construction based on lattices.

Bitranslucent sets. Informally, a *translucent set* is a subset P of a universe U, which can be sampled efficiently using only public information, and which is pseudorandom unless its associated secret key is known (in which case it is easily distinguishable from uniform over the universe).

Here we strengthen the notion of a translucent set in two ways. The first, main strengthening essentially preserves the pseudorandomness of P *even if the secret key is revealed.* Of course this is impossible as just stated, because the secret key makes P 'transparent' by design. Instead, we introduce an algorithm that, given a 'faking' key for the set system, and some c drawn from the pseudorandom set P, is able to *resample* a new, 'good-looking' secret key for which c appears uniformly random. We stress that such keys are necessarily very rare, because a typical secret key should correctly recognize P with high probability. Nevertheless, it can still be the case that for any $c \in P$, there are a few rare keys that misclassify c (without making it apparent that they are doing so). A bitranslucent set scheme is able to use the faking key find such keys.

Looking ahead to our bideniable encryption scheme, bitranslucency will allow a coerced sender and receiver both to plausibly claim that a value $c \in P$ is actually uniform: the sender simply declares that c was chosen uniformly from U (by claiming c itself as the random coins), and the receiver resamples a secret key that also makes c appear uniform.

The second strengthening, which yields qualitative improvements in efficiency and may have independent applications, allows for multiple translucent sets to share a single, fixed-size faking key. Essentially, this makes the bitranslucent set 'identity-based' (although we do not refer to identities explicitly): each translucent set has its own public and secret keys for generating and distinguishing P- and U-samples, and the master faking key makes it possible to generate a good-looking secret key that equivocates a given P-sample as a U-sample. Interestingly, this implies a bideniable encryption scheme in which the deniable key generator's faking key is a *fixed* size independent of the message length, despite the information-theoretic bound that *normal* secret keys must exceed the message length.

Definition 3 (Bitranslucent Set Scheme (BTS)). *A* bitranslucent set *scheme BTS is made up of the following algorithms:*

- *The* normal setup *procedure* $\mathsf{Setup}(1^n; r_{\mathsf{Setup}})$ *outputs a public parameter pp. We require that* Setup *has invertible sampling via an algorithm* I_{Setup}.
- *The* deniable setup *procedure* $\mathsf{DenSetup}(1^n)$ *outputs a public parameter pp and a* faking key fk.
- *The* key generator $\mathsf{Gen}(pp; r_R)$ *outputs a public key pk, whose associated secret key is the randomness* r_R *(without loss of generality).*
  We require that Gen *has trapdoor invertible sampling via an algorithm* I_{Gen} *and trapdoor* fk*, where* $(pp, fk) \leftarrow \mathsf{DenSetup}(1^n)$.
- *The P- and U-*samplers $\mathsf{SampleP}(pp, pk; r_S)$ *and* $\mathsf{SampleU}(pp, pk; r_S)$ *each output some* $c \in \{0,1\}^*$.
- *The* P-tester $\mathsf{TestP}(pp, r_R, c)$ *either accepts or rejects.*
- *The* sender-coins faker $\mathsf{FakeSCoins}(pp, pk, r_S)$ *outputs some coins* r_S^* *for the U-sampler.*[2]
- *The* receiver-coins faker $\mathsf{FakeRCoins}(pp, fk, pk, c)$ *outputs some coins* r_R^* *for the key generator.*

We require:

1. (Correctness.) *The following experiment should accept (respectively, reject) with overwhelming probability over all its randomness: let* $pp \leftarrow \mathsf{Setup}(1^n)$, $pk \leftarrow \mathsf{Gen}(pp; r_R)$, $c \leftarrow \mathsf{SampleP}(pp, pk; r_S)$ *(resp.,* $c \leftarrow \mathsf{SampleU}(pp, pk; r_S)$*), and output* $\mathsf{TestP}(pp, r_R, c)$.
   We also require correctness for the faking algorithms: with overwhelming probability over all the random choices, letting $(pp, fk) \leftarrow \mathsf{DenSetup}(1^n)$ *and* $pk \leftarrow \mathsf{Gen}(pp; r_R)$*, we should have*

$$\mathsf{SampleU}(pp, pk; \mathsf{FakeSCoins}(pk, r_S)) = \mathsf{SampleP}(pp, pk; r_S)$$
$$\mathsf{Gen}(pp; \mathsf{FakeRCoins}(pp, fk, pk, \mathsf{SampleP}(pp, pk; r_S))) = pk.$$

[2] In some realizations, including our own, $\mathsf{FakeSCoins}$ can directly compute coins r_S^* given just a ciphertext $c \leftarrow \mathsf{SampleP}(pp, pk; r_S)$, not r_S (or even pp, pk). We retain the more relaxed definition above for generality.

2. (Indistinguishable public parameters.) *The public parameters pp as produced by the two setup procedures $pp \leftarrow \mathsf{Setup}(1^n; r_{\mathsf{Setup}})$ and $(pp, fk) \leftarrow \mathsf{DenSetup}(1^n)$ should be indistinguishable (either statistically or computationally).*
3. (Bideniability.) *The following two experiments should be computationally indistinguishable:*

$(pp, fk) \leftarrow \mathsf{DenSetup}(1^n)$	$(pp, fk) \leftarrow \mathsf{DenSetup}(1^n)$
$pk \leftarrow \mathsf{Gen}(pp; r_R)$	$pk \leftarrow \mathsf{Gen}(pp; r_R)$
$c \leftarrow \mathsf{SampleU}(pp, pk; r_S)$	$c \leftarrow \mathsf{SampleP}(pp, pk; r_S)$
	$r_R^* \leftarrow \mathsf{FakeRCoins}(pp, fk, c)$
	$r_S^* \leftarrow \mathsf{FakeSCoins}(pk, r_S)$
Return (pp, r_R, r_S)	*Return* (pp, r_R^*, r_S^*)

Remark 1. For correctness, it suffices (and is more convenient) to require that when $c \leftarrow \mathsf{SampleU}(pp, pk)$, $\mathsf{TestP}(pp, r_R, c)$ just rejects with probability at least (say) $1/2$. The error probability can then be made negligible by parallel repetition of Gen and $\mathsf{SampleU}$, using the same public parameters pp.

The definition is designed to allow for the use of many public keys pk_i and associated secret keys $r_{R,i}$ under the same public parameter pp. This lets the sender and receiver exchange multiple bits more efficiently, and have shorter keys/randomness, than if the entire system were parallelized. In addition, in deniable mode, the receiver can perform all its secret-key operations (for multiple public keys pk_i) using just the single faking key fk, without keeping any of the random strings $r_{R,i}$ that were used to generate the pk_i, by just resampling $r_{R,i}$ as needed using $I_{\mathsf{Gen}}(fk, pp, pk_i)$. This trivially allows the receiver to decrypt, and to open P-samples and U-samples 'honestly' (without equivocation), in deniable mode.

Note that in the bideniability property above, both experiments use the *deniable* setup algorithm $\mathsf{DenSetup}$, rather than using the normal setup in the left-hand experiment. However, the choice of setup in the left experiment is essentially arbitrary, and the definition would be equivalent if we replaced the first line of the left-hand experiment with a normal setup $pp \leftarrow \mathsf{Setup}(1^n; r_{\mathsf{Setup}})$. This is because the faking key fk is never used, the public parameters are indistinguishable, and Setup has invertible sampling. We chose the presentation above because it yields a more modular proof that one can securely equivocate many independent public key/ciphertext pairs (pk_i, c_i) under a single public parameter pp: first, the bideniability property allows us to replace each pair of calls to the faking algorithms, one by one, with their normal counterparts. Then, finally, the deniable setup can be replaced with a normal setup, as just described.

Construction of deniable encryption. Canetti *et al.* [10] described a simple encoding trick to construct a multi-distributional sender-deniable encryption scheme from a translucent set: the normal encryption algorithm encodes a 0 message as "UU" whereas the deniable encryption algorithm encodes it as "PP;" both algorithms encode 1 as "UP." Thus, the sender can always open a deniably generated ciphertext as any message bit, by equivocating zero or more Ps as Us.

The same encoding lets us construct a multi-distributional *bideniable* encryption scheme from a bitranslucent set, since now the sender and receiver are both able to produce 'fake' coins that simultaneously equivocate a P as a U. Using the security properties of BTS, the proof of the following theorem (which we given in the full version) is routine.

Theorem 3. *Existence of a bitranslucent set scheme implies existence of a bideniable encryption scheme, secure under chosen-plaintext attacks.*

Canetti *et al.* [10] also construct a *fully* sender-deniable scheme from a translucent set, where the 'fake' coins can be distinguished with an inverse-polynomial probability related to the public key size. We show that for bideniability an analogous construction from a bitranslucent set also works, but requires the sender and receiver to 'coordinate' their fake coins as described Section 4. However, this coordination can again be removed using simulatable encryption. Details are deferred to the full version.

6 Lattice-Based Bitranslucent Set

In this section we give an overview of a bideniable translucent set scheme based on lattice problems, and the intuition behind its security. Due to the large amount of background required to formalize the system, space restrictions require us to defer a complete description and security proof to the full version.

Here we describe the main ideas behind our construction. To start, it will help to consider a basic mechanism for a (sender-deniable) translucent set. The public key is the description of a lattice Λ, and the secret key is a *short* lattice vector $\mathbf{z} \in \Lambda$. (A lattice is a periodic "grid" of points — formally, a discrete additive subgroup — in $\mathbb{R}^m$.) A P-sample is a point very close to the *dual lattice* Λ^*, while a U-sample is a uniformly random point in a certain large region. (The dual lattice Λ^* is the set of points $\mathbf{v} \in \text{span}(\Lambda)$ for which $\langle \mathbf{z}, \mathbf{v} \rangle \in \mathbb{Z}$ for every $\mathbf{z} \in \Lambda$.) With knowledge of $\mathbf{z}$, it is possible to distinguish these two types of points $\mathbf{c}$ by computing the inner product $\langle \mathbf{z}, \mathbf{c} \rangle$: when $\mathbf{c}$ is close to some $\mathbf{v} \in \Lambda^*$, the inner product $\langle \mathbf{z}, \mathbf{c} \rangle \approx \langle \mathbf{z}, \mathbf{v} \rangle \in \mathbb{Z}$ is close to an integer. On the other hand, when $\mathbf{c}$ is uniform, the inner product is uniformly random modulo $\mathbb{Z}$. Moreover, distinguishing between P-samples and U-samples (given only the public information) is essentially the decision-LWE problem, when the lattice Λ is defined appropriately in relation to the LWE instance. All of this is so far very similar to the structure of lattice-based encryption going back to the seminal scheme of Ajtai and Dwork [4], but in that system, the public key uniquely defines a secret key while the ciphertext does not uniquely define the encryption randomness. By contrast, here we have essentially described the 'dual' cryptosystem of Gentry, Peikert, and Vaikuntanathan [21], where there are many short vectors in Λ and hence many valid secret keys, and the ciphertext uniquely specifies the encryption randomness.

To construct a *bitranslucent* set, we exploit and build upon additional techniques from [21]. The faking key for the scheme is a *short basis* $\mathbf{T}$ of Λ (constructed using techniques from [3, 6]). As shown in [21], such a basis acts as a 'master trapdoor' that allows for efficiently sampling secret keys under a Gaussian-like distribution, and for efficiently extracting the short offset vector $\mathbf{x}$ from a P-sample $\mathbf{c} = \mathbf{v} + \mathbf{x}$, where $\mathbf{v} \in \Lambda^*$.

Using the above facts, our receiver faking algorithm works as follows: given the short basis $\mathbf{T}$ and a P-sample $\mathbf{c} = \mathbf{v} + \mathbf{x}$ that needs to be 'faked' as a U-sample, the algorithm first extracts $\mathbf{x}$, and then samples a secret key $\mathbf{z}^* \in \Lambda$ that is *highly correlated* with $\mathbf{x}$. By this we mean that $\mathbf{z}^*$ comes from a Gaussian distribution (over Λ) centered at $u \cdot \mathbf{x}$, for some random and large enough correlation coefficient $u \in \mathbb{R}$. Modulo $\mathbb{Z}$, the inner product $\langle \mathbf{z}^*, \mathbf{c} \rangle \approx \langle \mathbf{z}^*, \mathbf{x} \rangle \approx u \cdot \|\mathbf{x}\|^2$, because $\mathbf{z}^* \in \Lambda$ is short. When u is chosen from a wide enough Gaussian distribution, this inner product is uniform (modulo $\mathbb{Z}$), hence $\mathbf{c}$ "looks like" a U-sample when tested with the fake secret key $\mathbf{z}^*$. The natural question at this point is, why should it be secure to release a $\mathbf{z}^*$ that is so correlated with the secret offset vector $\mathbf{x}$? That is, why do $\mathbf{z}^*$ and $\mathbf{c}$ 'look like' an honestly generated secret key and U-sample, respectively? The first key point is that as Gaussian random variables, the correlation between $\mathbf{x}$ and $\mathbf{z}^*$ is essentially *symmetric*. That is, we can consider an alternative experiment in which $\mathbf{z}^*$ is chosen first (according to the normal key generation algorithm), and then $\mathbf{x}$ is chosen from a Gaussian centered at $v \cdot \mathbf{z}^*$ (for some appropriate random $v \in \mathbb{R}$). In both experiments, the pair $(\mathbf{z}^*, \mathbf{x})$ is jointly distributed as a nonspherical Gaussian, with the same covariance. This allows us to switch from the faking experiment to one in which the 'faked' secret key $\mathbf{z}^*$ is generated normally, followed by $\mathbf{c} = \mathbf{v} + \mathbf{x}$ where $\mathbf{v} \in \Lambda^*$ and $\mathbf{x}$ is centered at $v \cdot \mathbf{z}^*$. The second key point is that when $\mathbf{x}$ is correlated with $\mathbf{z}^*$ as described above, it may be written as the sum of two terms: the component $v \cdot \mathbf{z}^*$, and an independent (spherical) Gaussian component $\mathbf{g}$. Under the LWE assumption, we can switch from $\mathbf{c} = (\mathbf{v} + \mathbf{g}) + v \cdot \mathbf{z}^*$ to $\mathbf{c}' = \mathbf{u} + v \cdot \mathbf{z}^*$, where $\mathbf{u}$ is uniformly random. Of course, the latter quantity $\mathbf{c}'$ is truly uniform and independent of the (normally generated) secret key $\mathbf{z}^*$. This coincides exactly with the generation and subsequent honest opening of a normal secret key and U-sample, as desired.

Acknowledgments. We thank Ran Canetti and Cynthia Dwork for suggesting this research topic, Josh Benaloh and Ari Juels for helpful discussions about deniability, and the anonymous reviewers for useful comments.

References

[1] The rubberhose encryption system. Internet website (accessed February 9, 2010), `http://iq.org/~proff/marutukku.org/`

[2] Truecrypt: Free open-source on-the-fly encryption. Internet website (accessed Feburary 9, 2010), `http://www.truecrypt.org`

[3] Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 1–9. Springer, Heidelberg (1999)

[4] Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: STOC, pp. 284–293 (1997)

[5] Alwen, J., Dodis, Y., Wichs, D.: Survey: Leakage resilience and the bounded retrieval model. In: Kurosawa, K. (ed.) Information Theoretic Security. LNCS, vol. 5973, pp. 1–18. Springer, Heidelberg (2010)
[6] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS, pp. 75–86 (2009)
[7] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)
[8] Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
[9] Bendlin, R., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: Receiver-deniable public-key encryption is impossible. Cryptology ePrint Archive, Report 2011/046 (2011), `http://eprint.iacr.org/`
[10] Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
[11] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
[12] Canetti, R., Gennaro, R.: Incoercible multiparty computation (extended abstract). In: FOCS, pp. 504–513 (1996)
[13] Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. J. Cryptology 20(3), 265–294 (2007), Preliminary version in EUROCRYPT 2003.
[14] Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
[15] Chvátal, V.: The tail of the hypergeometric distribution. Discrete Math. 25, 285–287 (1979)
[16] Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
[17] Duermuth, M., Freeman, D.M.: Deniable encryption with negligible detection probability: An interactive construction. Cryptology ePrint Archive, Report 2011/066 (2011), `http://eprint.iacr.org/`
[18] Dürmuth, M., Freeman, D.M.: Deniable encryption with negligible detection probability: An interactive construction. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 610–626. Springer, Heidelberg (2011)
[19] Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. J. ACM 50(6), 852–921 (2003); Preliminary version in FOCS 1999
[20] Garay, J.A., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 505–523. Springer, Heidelberg (2009)
[21] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)
[22] Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
[23] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 1–40 (2009); Preliminary version in STOC 2005
[24] Wikipedia. Deniable encryption — Wikipedia, the free encyclopedia. Internet website (2010), `http://en.wikipedia.org/wiki/Deniable_encryption` (accessed February 9, 2010)