A preliminary version of this paper appears in *Proceedings of the 13th Annual Conference on Computer and Communications Security*, ACM, 2006. This is the full version.

Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation

Mihir Bellare* Tadayoshi Kohno† Victor Shoup‡

August 2006

Abstract

We show how to significantly speed-up the encryption portion of some public-key cryptosystems by the simple expedient of allowing a sender to maintain state that is re-used across different encryptions. In particular we present stateful versions of the DHIES and Kurosawa-Desmedt schemes that each use only one exponentiation to encrypt, as opposed to two and three respectively in the original schemes, yielding the fastest discrete-log based public-key encryption schemes known in the random-oracle and standard models respectively. The schemes are proven to meet an appropriate extension of the standard definition of IND-CCA security that takes into account novel types of attacks possible in the stateful setting.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`. Supported in part by NSF grant CNS-0524765 and a gift from Intel Corporation.

†Dept. of Computer Science & Engineering, University of Washington, AC101 Paul G. Allen Center, Box 352350, 185 Stevens Way, Seattle, WA 98195-2350. E-mail: `yoshi@cs.washington.edu`. URL: `http://www.cs.washington.edu/homes/yoshi`. Work done while at UCSD, supported in part by grants of first author.

‡Dept. of Computer Science, New York University, Courant Institute, 251 Mercer Street, New York, NY 10012, USA. E-mail: `shoup@cs.nyu.edu`. URL: `http://www.shoup.net`. Supported in part by NSF grant CCR-0310297.

Contents

1 Introduction

The discrete-exponentiation computations underlying public-key cryptography are expensive, in particular having a cost thousands of times that of blockcipher or hash function computations. This not only results in slowdown, but, on systems of limited computing power such as sensors, can even be a barrier to adding public-key cryptography at all. Today, the cost of public-key cryptography is being felt even more acutely due to new resource constraints on emerging computing platforms: public-key cryptography operations are a severe drain on power and reduce battery life, which is the main limitation on mobile devices such as cell phones, PDAs, RFID chips and sensors. For these and more well-known reasons, there is an obvious interest in reducing the number of discrete exponentiations required for public-key cryptography. This is a domain where a 10% improvement would be very welcome and a 50% improvement would be dramatic.

Much work goes into improved algorithms, time-space tradeoffs (pre-computation) and faster implementations for discrete exponentiation. However these methods have been pushed pretty much as far as they can go, and bring only minor improvements at this point. Our approach is different. We propose to change the *model* for public-key encryption. The change is simple, namely that we allow senders to be stateful, maintaining information that they re-use across different encryptions. We will show that this is highly effective by presenting stateful encryption schemes whose encryption time beats that of all known (stateless) schemes by margins of 50% or more without impacting decryption time or requiring extra storage. We will show that these gains are not at the cost of assurance by providing security definitions and proofs to support our designs. We will also explain why stateful encryption is convenient to implement and deploy in systems so that the cost benefits obtained are realizable in practice. Let us now look at all this in some more detail.

Background and state of the art. In the standard model of public-key encryption [26], the sender is stateless (i.e. memoryless), encrypting each message as a function (only) of the message, recipient's public key and freshly-chosen coins. Due to its being what is needed for the security of encryption-using applications [37], security against chosen-ciphertext attack (IND-CCA) [34, 22, 9] is the accepted security goal. We are interested in discrete-logarithm based, proven-IND-CCA schemes. (We discuss RSA later.) The most efficient such scheme is DHIES [2], an ElGamal-based scheme where encryption and decryption cost two and one exponentiations respectively.

The new model. In the new model we propose, the sender maintains state information. The encryption function applied by the sender takes not only the message, receiver's public-key and freshly-chosen coins, but also the current state, returning a ciphertext and a possibly updated state that replaces the previous one. Decryption is unchanged from a standard stateless system, and depends only on the ciphertext and secret key of the receiver. (The receiver is not stateful, and need not even know that the sender is.) Why this simple change to the model should result in (much) faster encryption may not be clear, but we will now see that it is so.

Stateful DH. We present a stateful variant of DHIES that drops the encryption cost to one exponentiation without increasing decryption cost. The idea is simple. Recall that the ElGamal encryption of message M under public key g^x has the form $(g^r, g^{xr} \cdot M)$ where r is the random coins chosen by the sender, anew for each encryption. It is natural to try to use (r, g^r) as the state, meaning re-use r across different encryptions so that g^r does not need to be computed each time, but this is clearly insecure, in particular because the resulting scheme is deterministic. However, if instead we derive a symmetric key from g^{xr} via a hash function, and send as ciphertext g^r together with a symmetric encryption of M under a (randomized, stateless) IND-CCA secure symmetric scheme, then the scheme can be shown to be secure. Note that symmetric schemes of the required

Scheme	Cost		Security	
	Enc	**Dec**	**Assumption**	**RO?**
DHIES [2]	2	1	Gap-DH	Yes
StDH	1	1	Gap-DH	Yes
KD [30]	3	1	DDH	No
StKD	1	1	DDH	No

Figure 1: DHIES and KD compared to our stateful variants StDH and StKD. We show the number of exponentiations for encryption and decryption as well as the assumptions for the proofs of security and whether or not it is in the RO model. Note all these schemes are hybrid, but for simplicity the table does not show the (negligible) costs and (standard) assumptions related to the symmetric components of the schemes.

---

type are easily and cheaply obtained [8, 10]: one can for example use an AES mode of operation like CBC in encrypt-then-mac combination with a secure MAC like HMAC [7] or CMAC [32]. An interesting feature of our scheme is the crucial use made of hybrid encryption, namely the combination of asymmetric and symmetric primitives.

Stateful KD. It is common in cryptography that the security proofs of the most efficient proven-secure schemes are in the random-oracle (RO) model [13]. This is the case for DHIES and its stateful variant StDH discussed above. Concerns about the difficulty of instantiating ROs [18, 5] have lead to interest in standard-model schemes. In this domain, the most efficient known (stateless) IND-CCA scheme is the Kurosawa-Desmedt [30] variant KD of the Cramer-Shoup [21] scheme, where encryption and decryption cost three and one exponentiations respectively. We present a stateful variant of KD which needs *just one* exponentiation each for encryption and decryption, just as for StDH. Remarkably, not only is StKD the first non-RO scheme that is as efficient as RO ones, but also it is more efficient than any previous (stateless) schemes, whether with ROs or not!

Instantiation. The preferred choice of group is an elliptic-curve one, where the discrete logarithm problem is already hard for 160-bits. This minimizes exponentiation time and ciphertext size.

Variants. In StDH and StKD, encryption does not even modify the current state, which stays of constant size. If one is willing to modify and grow the state with encryption, further optimizations are possible, taking advantage of the fact that both StDH and StKD encrypt symmetrically under a key that is a deterministic function of the state and recipient public key. By caching this key in the state the first time it is computed, subsequent encryptions to the same recipient require only symmetric operations.

Crash-robustness and state-reset. A sender-side system or application crash is a concern for stateful schemes because the current state would be lost. The danger this poses is exemplified with a typical stateful symmetric encryption scheme such as counter-mode with zero-initialized counter. If the current counter value is lost and encryption restarts with a re-initialized counter, privacy is compromised. Not so in our schemes, which are robust in the face of crashes. The sender can pick a new initial state (i.e. reset its state) and restart, and security is maintained. It is safe to pick a new state when a system reboots, and safe to have different concurrent applications each handling its own state. Resetting state does have a computational cost, but in our schemes this is exactly the difference in cost between the stateless and stateful versions. If you reset rarely (as we imagine will usually be the case) your cost is that of the stateful scheme. In the worst case that you reset for each and every encryption, you only return to the cost of the stateless scheme.

Maintaining state. Maintaining state is not difficult on a system, particularly given the robustness discussed above. Indeed, even though the current model of encryption is stateless at the mathematical level, systems will typically use state in implementing it, in the form of a seed from which the encryption algorithm's random choices are pseudorandomly generated. One issue is that privacy is lost if the state is compromised, but this can be addressed by not too infrequent (eg. once a day) state resets.

What about pre-computation? The above-quoted factor of two or more performance improvement of our schemes compared to previous ones does not take into account pre-computation based speed-up. The bottom-line is that while pre-computation narrows the gap it does not eliminate it. Most importantly, the stateful schemes achieve the greater efficiency *without* any of the storage cost associated to pre-computation, which is attractive for platforms which are storage-limited. To elaborate, recall that exponentiation to a fixed base can be made a factor w faster by pre-computing and storing a table of $(160/w)2^w$ appropriate powers of the base. This can be applied to one of the two exponentiations underlying DHIES (the base is not fixed for the other one) and not at all to StDH (where the base is not fixed). If we set w, to, say, 5, the ratio of the cost of DHIES to the cost of StDH drops from 2 to 1.2. However, not only is a 20% improvement not to be sneezed at, but StDH achieves this without the need to store the $160 \cdot (160/5) \cdot 2^5 = 163,840$ bit table needed to speed-up DHIES, so that, overall, statefulness remains a win.

What about RSA? Encryption is already fast in RSA because we can use a small encryption exponent like 3, but one needs a 1024-bit modulus to get the same security as elliptic-curve discrete-logarithm based schemes offer with 160-bit groups. The result is that RSA decryption, even though a single exponentiation, is slower than in the discrete-log based schemes. Also, ciphertexts are larger. Thus the discrete-log based systems are more attractive in many settings, particularly if encryption time is dropped as in our stateful schemes.

Security Proofs. In introducing a new model and schemes, there is the danger of having also introduced new security vulnerabilities. The high cost associated to software or hardware updates resulting from bugs in deployed cryptography means that we want to address this by providing pre-implementation security assurance. The most convincing form for this is a proof that the schemes meet an appropriate, well-defined notion of security. This is what we provide.

Security definition. We begin with a threat-analysis that identifies paths for attack not covered by the classical definition of IND-CCA security for stateless schemes [34, 22, 9]. Briefly, there are two main issues. The first is that encryption now depends on a quantity not a priori known to the adversary, namely the sender's state, and so the ability of an adversary to see ciphertexts is no longer captured merely by giving it the public key as in [26, 34, 22]. We address this by giving the adversary an oracle for encryption under the sender's state. The second issue is that encryptions sent by one sender to different receivers, being computed using the same or related state, are not independent of each other, and so an adversary might be able to compromise privacy of messages encrypted under one public key by seeing ciphertexts of related messages encrypted by the same sender under a different, *maliciously chosen* public key. Thus we need to consider attacks in which the adversary is allowed to choose public keys for malicious receivers and see ciphertexts that a sender encrypted under these keysin an attempt to determine information about a ciphertext that the same sender encrypted to an honest recipient. A definition taking all these issues into account is provided in Section 3, and, above and below, when we refer to a stateful scheme being IND-CCA, we mean that it meets this definition.

Proofs for our schemes. We prove StDH is IND-CCA secure in the RO model assuming Gap-DH. (The Gap-DH assumption, due to Okamoto and Pointcheval [33], says that CDH remains hard even given an oracle for DDH.) We prove StKD is IND-CCA secure assuming, as in [30, 25], that the DDH (Decision Diffie-Hellman) problem is hard. In both cases we also make assumptions on the security of the symmetric encryption schemes used. These are slightly stronger than the ones made in the original papers [2, 25], but symmetric schemes satisfying these assumptions are easily and efficiently built via blockcipher modes of operation and MACs.

However there is one difference between what is proven about StDH and StKD. Recall that in our model the adversary can choose public keys and ask the sender to encrypt under them. For StKD, we only let the adversary provide public keys if it knows the corresponding secret key. (In the formal model, which we call the known secret key model and was first used in [6, 15], we simply require it to provide the secret key.) This reflects the assumption that the CA requires a proof of possession of a secret key corresponding to any public key it certifies. Removing this assumption without invoking ROs is an interesting open problem.

Related work. The multi-recipient encryption schemes of Kurosawa [29] and Bellare, Boldyreva and Staddon [6] allow a sender to batch-encrypt n messages to n different encryptions at a cost lower than that of n separate encryptions. For example, their ElGamal based scheme uses for this task only $n+1$ exponentiations rather than the naive $2n$. However their schemes are limited, requiring that the recipients have to all be different, meaning have different public keys, and requiring that the sender have all the public keys and messages together and process them as a batch. Our work is an extension of theirs. Our model is more flexible. Once an initial state is chosen, we can process any number of on-line encryption requests, these to any recipient. Repeat encryptions to a single recipient are allowed. Yet we pay only one exponentiation per encryption. Our schemes use the randomness-recycling idea of [29, 6] but differ from theirs in the crucial use of hybrid encryption. Note that our schemes yield new multi-recipient encryption schemes, in particular, via StDH, the first one that does not use the known secret key model. (All the schemes of [29, 6] use this model.)

In the signcryption model [38, 4, 3] where both the sender and receiver have (certified) public keys, say g^r, g^x respectively, one can encrypt (and authenticate) cheaply under the implicitly shared key g^{rx}. In our setting however the sender does not need a public key or certificate. (And we are interested only in privacy.) This is the more pragmatic setting, reflecting a typical Internet SSL connection, where the server has a certificate but the client does not.

IBE-based approaches have been yielding efficient IND-CCA public-key encryption schemes in the standard model [19, 14, 28]. It is interesting to ask whether these schemes have stateful variants with lower encryption cost.

As noted in Figure 1, DHIES too can be proved under the Gap-DH assumption in the RO model. However, DHIES is also proved IND-CCA secure in [2] in the standard model under an assumption they call Oracle-DH (ODH). StDH can be proved under this assumption too, thereby avoiding the RO model. But ODH is a non-standard assumption mixing a hash function and number-theory.

Stateful encryption is well-known in the symmetric setting with schemes like counter mode encryption, analyzed in [8]. Our work extends this to the asymmetric setting.

2 Notation

A string means a binary one. If x, y are strings, then $|x|$ is the length of x. If S is a finite set, then $|S|$ is its cardinality, and $s \stackrel{\$}{\leftarrow} S$ denotes the operation of assigning to s an element of S chosen at random. If A is a randomized algorithm, then $y \stackrel{\$}{\leftarrow} A(x_1, \ldots)$ denotes the operation (experiment) of running A on inputs $x_1, \ldots$, and letting y denote the output.

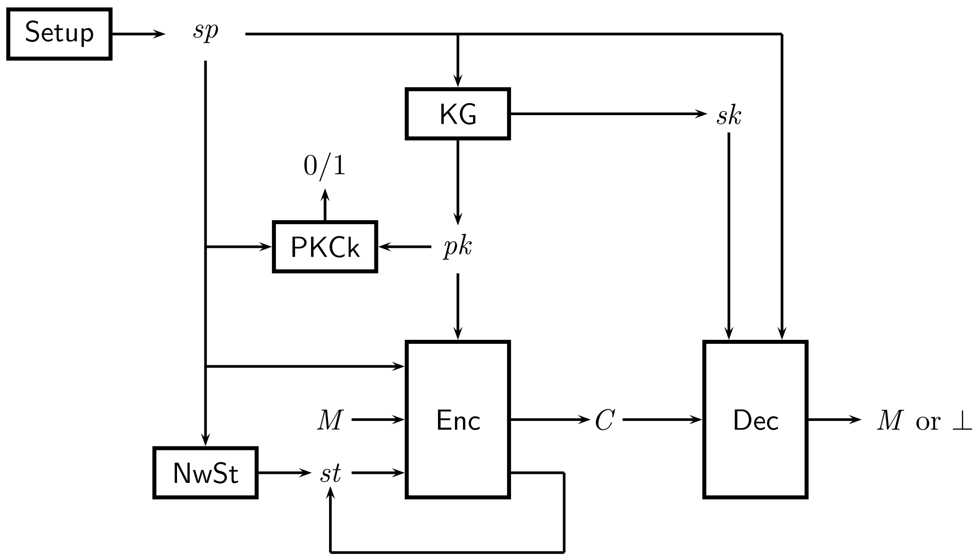

Figure 2: Components and operation of a stateful public-key encryption scheme.

3 Stateful encryption schemes

Syntax and operation. A *stateful public-key encryption scheme* $\mathsf{StPE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{PKCk}, \mathsf{NwSt}, \mathsf{Enc}, \mathsf{Dec})$ is specified by six algorithms (all possibly randomized except the last) whose operation is illustrated in Figure 2. The setup algorithm Setup is run by an authority to produce system parameters sp. Any entity that wants to receive encrypted data can run the key-generation algorithm KG on input sp to get a public and secret key for itself. A sender maintains state st whose initial value is obtained by running the new state algorithm NwSt on input sp. At any time, the sender can apply the encryption algorithm to sp, a public key pk, message M and its current state st to get two outputs. The first, C, is a ciphertext for the receiver whose public key is pk, and the second is an updated value for the state variable for the sender. We write this algorithmically as $(C, st) \xleftarrow{\$} \mathsf{Enc}(sp, pk, M, st)$. At any time, the sender can refresh its state by again running NwSt, which is important for robustness in the face of state-loss due to system failures as discussed in Section 1. The deterministic decryption algorithm Dec, run by the receiver, takes input sp, the receiver's secret key sk and a ciphertext C and returns either a message M or $\perp$ to indicate that the ciphertext is invalid.

The security of our schemes requires that the sender avoid encrypting unless the given public key passes some simple check, which in our case just involves testing that certain components of the key are members of the underlying group. Such a group-membership check could either be done by the encryption algorithm or by a CA as a condition to issuing a certificate for the public key. (Implicitly we are assuming that encryption is not done unless the sender has a valid certificate for the receiver's public key, just as for standard public-key encryption.) Which is better would depend on the cost or other things. In any case, since such a check could be placed in several places, we have represented it in the syntax via a separate public-key verification algorithm PKCk that given sp and a string pk returns a bit representing the outcome of the check on candidate public key pk. Note we are talking only about very simple checks, like whether components of the public key are members of the underlying group, not whether the supplier of the public key "knows" a corresponding secret key, which we will discuss below in the context of security. Simple checks like this (that things are group elements) are sometimes required for the security of standard schemes as well but not made explicit [17, 2].

Note that the sender uses the same state for communicating with different receivers who have different public keys, and the generation of a fresh state requires only sp and no information about potential receivers. In our main schemes, the encryption algorithm does not even modify the state, meaning that the output state is the same as the input one and in particular has constant size, but we will see variants in which it does. Different senders maintain different states. Receivers are not stateful, and in particular there is thus no state synchronization issue.

We require a natural consistency condition, which says that for any sp generated by Setup, any $n \geq 1$, and any keys $(pk_1, sk_1), \ldots, (pk_n, sk_n)$ generated via $\mathsf{KG}(sp)$, if st_0 is an output of $\mathsf{NwSt}(sp)$ and (C_i, st_i) is an output of $\mathsf{Enc}(sp, pk_i, M_i, st_{i-1})$ for all $i \in \{1, \ldots, n\}$, then $\mathsf{Dec}(sk_i, C_i) = M_i$ for all $i \in \{1, \ldots, n\}$.

Basic security model. To define IND-CCA security of stateful public-key encryption scheme $\mathsf{StPE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{PKCk}, \mathsf{NwSt}, \mathsf{Enc}, \mathsf{Dec})$ we consider a game played with an adversary A. The game begins with the initializations

$$sp \xleftarrow{\$} \mathsf{Setup} \; ; \; (pk_1, sk_1) \xleftarrow{\$} \mathsf{KG}(sp) \; ; \; c \xleftarrow{\$} \{0,1\} \; ; \; n \leftarrow 1 \; ; \; st \xleftarrow{\$} \mathsf{NwSt}(sp)$$

to generate system parameters, a key-pair for a (single) honest recipient, and a challenge bit c. Adversary A is then given input sp, pk_1 and allowed to query various oracles, after which it outputs a bit. The following shows what types of oracle queries A can make, what action the game takes in each case, and what response the game returns to A:

Oracle query	**Action taken by game**	**Response**
$\textsc{MkBdRec}(pk)$	If $\mathsf{PKCk}(sp, pk) = 1$ then $n \leftarrow n+1$; $pk_n \leftarrow pk$	n
$\mathrm{LOR}(M_0, M_1)$	$(C, st) \xleftarrow{\$} \mathsf{Enc}(sp, pk_1, M_c, st)$	C
$\textsc{Encrypt}(i, M)$	$(C, st) \xleftarrow{\$} \mathsf{Enc}(sp, pk_i, M, st)$	C
$\textsc{Decrypt}(C)$	$M \leftarrow \mathsf{Dec}(sp, sk_1, C)$	M

The game is modeling a (single) stateful sender communicating with multiple receivers. Only one receiver, who in the game has index 1, is honest, having keys generated via KG. A controls the public keys of all other receivers: via $\textsc{MkBdRec}(pk)$, it can create a receiver with whose public key is any string pk of A's choice that passes the public-key verification test. (This models the fact that encryption will not be performed under a public key that does not pass the test, either because the CA would not certify it or because the encryption algorithm would refuse to do the test, depending on where we place the check in the scheme.) A can call LOR with arguments M_0, M_1 of its choice, with the restriction that $|M_0| = |M_1|$, to have the sender encrypt challenge message M_c, under the sender's current state and the public key of the honest receiver, and return the resulting ciphertext to A. In this game, we restrict A to at most one LOR query in its entire execution. (We will see below that this is wlog, and it simplifies proofs.) A can call $\textsc{Encrypt}(i, M)$, with the restriction that $1 \leq i \leq n$, to have the sender encrypt M, under the sender's current state and the public key of receiver i, and return the ciphertext to A. (Encryptions to different users are not independent since they depend on a common state. This query is to capture the possibility that encryption under maliciously chosen public keys leaks state information compromising encryption under the honest public key.) The $\textsc{Decrypt}$ oracle allows a chosen-ciphertext attack on the honest receiver, and of course A can call $\textsc{Decrypt}(C)$ only if C has not been previously returned in response to a LOR query. At the end of its execution, A must output a bit. Denoting it by d, the ind-cca-advantage of A is

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{StPE}}(A) = 2 \cdot \Pr\left[d = c\right] - 1 \, .$$

The above reflects the *unknown secret key* (USK) model in which the CA is not assumed to require a proof of knowledge of the secret key corresponding to a registered public key. In the *known secret key* (KSK) model, where this assumption is made, the only change is that the MkBdRec oracle takes not one but two arguments, pk and sk, and the adversary is restricted to calling this oracle with pk, sk which are possible outputs of $\mathsf{KG}(sp)$. (This way of abstracting out the proofs of knowledge begins with [6, 15] and leads to a simple and convenient model.) The action taken by the game is simply $n \leftarrow n+1$; $pk_n \leftarrow pk$, and the response returned is n. (The check as to whether $\mathsf{PKCk}(sp, pk) = 1$ is now redundant and thus omitted.) We will generally refer to the notion of security as IND-CCA in either case, clarifying whether it is the USK or KSK model as necessary.

An extended security model and equivalence with the basic one. The above model is a simplified one. It turns out security in this model implies many extra security features not explicit in the model. For example, it implies that the sender can reset its state at any time and retain security, which as we have seen is important to running a stateful system in practice. It also implies that, even if the current state is compromised, encryption under a reset state remains secure, so that resets can be used to recover from compromise. The basic model allows only a single sender, single honest receiver, and single LOR query, but one should ask whether allowing all these to be multiple would give the adversary more power. It does not, even if the adversary is allowed to dynamically corrupt receivers and obtain their secret keys, and reset or expose sender's states. In Appendix A we reflect all this. We define a richer, extended model which allows the extra capabilities we have discussed, and, via standard hybrid and guessing arguments, show that security in the extended model follows from security in the basic one. Specifically, the adversary's advantage changes by a factor that is at most polynomial in the number of oracle queries it makes. Given this, we use the basic model throughout the paper because doing so simplifies our proofs.

Random oracle model. For schemes in the random oracle (RO) model [13], as usual, any of the constituent algorithms may have access to a RO, and in the security experiment, the adversary gets access to it as well.

4 The stateful DH scheme

This section describes our stateful DH/ElGamal scheme StKD which is effectively a stateful version of the IND-CCA DHIES [2]. (Note ElGamal is only IND-CPA, not IND-CCA.)

Building blocks. We describe the building blocks used and assumptions made about them:

- A cyclic group $\mathbb{G}$ whose order is denoted m. Let $\mathsf{Gen}(\mathbb{G})$ denote the set of generators of $\mathbb{G}$. We assume the Gap-CDH problem [33] is hard in $\mathbb{G}$, captured by defining the gap-dh-advantage of an adversary $A_{\mathbb{G}}$ as

$$\mathbf{Adv}^{\text{gap-dh}}_{\mathbb{G}}(A_{\mathbb{G}}) = \Pr\left[\, Z = g^{xr} \;:\; g \xleftarrow{\$} \mathsf{Gen}(\mathbb{G}) \,;\, r, x \xleftarrow{\$} \mathbb{Z}_m \,;\, Z \xleftarrow{\$} A_{\mathbb{G}}^{\mathrm{DDH}_g(g^r,\cdot,\cdot)}(g, g^r, g^x) \,\right] ,$$

  where the *restricted DDH oracle* $\mathrm{DDH}_g(g^r, \cdot, \cdot)$, on input Y, W, returns 1 if $Y, W \in \mathbb{G}$ and $W = Y^r$, and 0 otherwise. This is weaker than a full DDH oracle because g^r is fixed, but the ensuing weaker assumption suffices for our results.
- A symmetric encryption scheme $\mathsf{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ with keylength k, specified by its (randomized) encryption algorithm SEnc and (deterministic) decryption algorithm SDec. It is assumed IND-CCA secure, captured by defining the ind-cca-advantage of an adversary A_{SE} as

$$\begin{aligned}&\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{SE}}(A_{\mathsf{SE}})\\ &= 2 \cdot \Pr\left[\, d = c \;:\; K \xleftarrow{\$} \{0,1\}^k \,;\, c \xleftarrow{\$} \{0,1\} \,;\, d \xleftarrow{\$} A_{\mathsf{SE}}^{\mathsf{SEnc}(K,\cdot),\mathrm{LOR}(K,\cdot,\cdot,c),\mathsf{SDec}(K,\cdot)} \,\right] - 1 .\end{aligned}$$

Above, $\mathrm{LOR}(K, M_0, M_1, c)$ returns $C_s \xleftarrow{\$} \mathsf{SEnc}(K, M_c)$. A_{SE} is allowed only one query to this left-or-right encryption oracle, consisting of a pair of equal-length messages, and is not allowed to query $\mathsf{SDec}(K, \cdot)$ on a ciphertext previously returned by the LOR oracle. (Thus we are using the find-then-guess notion from [8], formulated in a different but equivalent way.)

Scheme. The Setup algorithm of the *stateful DH scheme* $\mathsf{StDH} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{PKCk}, \mathsf{NwSt}, \mathsf{Enc}, \mathsf{Dec})$ returns as system parameters a generator g chosen at random from $\mathsf{Gen}(\mathbb{G})$. A receiver's secret and public keys, as created by $\mathsf{KG}(g)$, are (x, X) and X respectively, where x is chosen at random from $\mathbb{Z}_m$ and $X = g^x$. $\mathsf{PKCk}(g, X)$ simply verifies that $X \in \mathbb{G}$. The new state algorithm $\mathsf{NwSt}(g)$ picks r at random from $\mathbb{Z}_m$ and returns $st = (r, R)$ where $R = g^r$. Given system parameters g, public key X assumed to be in $\mathbb{G}$, message M, state $st = (r, R)$, and access to random oracle $H\colon \mathbb{G}^3 \to \{0,1\}^k$, encryption algorithm Enc computes the k-bit key $K = H(R, X, X^r)$ and symmetric ciphertext $C_s \xleftarrow{\$} \mathsf{SEnc}(K, M)$, returns (R, C_s) as the ciphertext, and returns an unmodified state (r, R). Note encryption in this scheme does not change the state, and *all* the randomness for the encryption (without which the scheme cannot be secure) comes from the symmetric encryption scheme. Given g, secret key (x, X), ciphertext $C = (R, C_s)$ and oracle H, decryption algorithm Dec returns $\perp$ if $R \notin \mathbb{G}$ and otherwise returns $\mathsf{SDec}(H(R, X, R^x), C_s)$ (which may be $\perp$).

Instantiations and cost. The cost for public-key operations is one exponentiation each for encryption and decryption, namely half and the same, respectively as for standard (stateless) DH/ElGamal type schemes such as DHIES [2]. State initialization costs one exponentiation, but this is expected to be done rarely. (And in the worst case that it is done before each encryption, the total cost only returns to that of the stateless scheme.) The most suitable group to use is a 160-bit elliptic curve one, in which case the cost for PKCk, namely a group membership test, is cheap enough to neglect. Note one can use a group with an efficiently computable pairing, in which case the assumption drops to just CDH since DDH is easy in such groups. (Interestingly, in this case, the pairing is used only by the algorithms in the proof of security and does not impact the performance of the scheme.) One can also use a group of integers modulo a prime or subgroup thereof, but there seems no point since the group will need to be larger, slowing everything down, and the check could cost an exponentiation. Ciphertext expansion (length of ciphertext minus length of plaintext) is 160 bits plus the expansion from the symmetric encryption.

For a symmetric IND-CCA scheme, the easiest is to use an encrypt-then-mac generic composition [10] with an AES mode of operation like CBC and a MAC like CBC-MAC or HMAC [7]. Using a dedicated authenticated encryption mode like OCB [35] will halve the cost. In both of these cases, we get a 128+80 bit expansion for the symmetric part. The expansion can be reduced at the cost of more block cipher operations by the schemes of [23]. But in any case, the symmetric costs are negligible compared to the asymmetric ones unless the message is very long.

The symmetric scheme needed in DHIES is slightly weaker than the one here because it only needs to be one-time IND-CCA. The computational cost of upgrading to IND-CCA is however negligible. But we do add 128 bits to the size of the ciphertext.

Variant. The symmetric key $H(R, X, X^r)$ could be computed only the first time the sender communicates with the receiver whose public key is X and then "cached" as part of the state, reducing the cost of further encryptions to this receiver to merely symmetric operations. Note in this scheme the encryption algorithm does modify the state.

Security of Stateful DH. The following implies that if the Gap-DH problem is hard in $\mathbb{G}$ and SE is IND-CCA secure then Stateful DH meets the notion of IND-CCA security for stateful schemes we defined previously, even in the USK model.

procedure Initialize

$x_1 \xleftarrow{\$} \mathbb{Z}_m$; $r \xleftarrow{\$} \mathbb{Z}_m$
$X_1 \leftarrow g^{x_1}$; $R \leftarrow g^r$
$c \xleftarrow{\$} \{0,1\}$
$n \leftarrow 1$
$K[R, X_1] \xleftarrow{\$} \{0,1\}^k$

procedure $\textsc{MkBdRec}(X)$

If $X \in \mathbb{G}$ then
    $n \leftarrow n+1$; $X_n \leftarrow X$
    If (not $K[R, X]$) then
        $K[R, X] \xleftarrow{\$} \{0,1\}^k$
Reply n

procedure $\textsc{Encrypt}(i, M)$

$C_s \xleftarrow{\$} \mathsf{SEnc}(K[R, X_i], M)$
Reply (R, C_s)

procedure $\mathrm{LOR}(M_0, M_1)$

$C_s \xleftarrow{\$} \mathsf{SEnc}(K[R, X_1], M_c)$
Reply (R, C_s)

procedure $\textsc{Decrypt}(C)$

Parse C as (S, C_s)
If $S \notin \mathbb{G}$ then reply $\perp$
If (not $K[S, X_1]$) then $K[S, X_1] \xleftarrow{\$} \{0,1\}^k$
Reply $\mathsf{SDec}(K[S, X_1], C_s)$

procedure $H(S, X, D)$

$H[S, X, D] \xleftarrow{\$} \{0,1\}^k$
If $S = R$ and $D = X^r$ then
    If $X \neq X_1$ then
        If (not $K[R, X]$) then $K[R, X] \xleftarrow{\$} \{0,1\}^k$
        $H[R, X, D] \leftarrow K[R, X]$
    Else
        $D^* \leftarrow D$
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $\boxed{H[R, X_1, D] \leftarrow K[R, X_1]}$
Reply $H[S, X, D]$

procedure Finalize(d)

If $c = d$ then return 1
Else return 0

Figure 3: Game G_0 includes the boxed statement, while game G_1 does not.

Theorem 4.1 *Let* StDH *be the stateful DH public-key encryption scheme associated to cyclic group* $\mathbb{G}$ *and symmetric encryption scheme* SE*. Let* A *be an ind-cca-adversary against* StDH *in the USK model. Then there exists a gap-dh-adversary* $A_{\mathbb{G}}$ *against* $\mathbb{G}$ *and a ind-cca-adversary* A_{SE} *against* SE *such that*

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{StDH}}(A) \;\leq\; 2 \cdot \mathbf{Adv}^{\text{gap-dh}}_{\mathbb{G}}(A_{\mathbb{G}}) + \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{SE}}(A_{\mathsf{SE}})\;. \tag{1}$$

Furthermore the running times of $A_{\mathbb{G}}, A_{\mathsf{SE}}$ *are that of* A *plus some overhead, in the first case for some computations of* $\mathsf{SEnc}, \mathsf{SDec}$*, and in the second for a number of exponentiations equal to the number of random-oracle queries of* A*.* ▮

Proof of Theorem 4.1: We use game-playing in the style of [11]. We first recall some of the notation and conventions from the latter used in the games of Figure 3. When adversary A interacts with game $G \in \{G_0, G_1\}$, first the **Initialize** procedure is executed. Then A is executed, its queries answered by the corresponding game procedures. The output bit d of A is the input to the **Finalize** procedure which then produces the game output, denoted G^A. The games use arrays $K[\cdot,\cdot], H[\cdot,\cdot,\cdot]$ assumed to be initially everywhere undefined, and a test of the form (not $K[S, X]$) returns true if $K[S, X]$ is undefined and false otherwise.

We assume wlog that A never repeats a query to the random oracle. We first claim the following:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{StDH}}(A) &= 2\Pr\left[G_0^A \Rightarrow 1\right] - 1 && (2)\\
&= 2 \cdot \left(\Pr\left[G_0^A \Rightarrow 1\right] - \Pr\left[G_1^A \Rightarrow 1\right]\right) + 2 \cdot \Pr\left[G_1^A \Rightarrow 1\right] - 1 && (3)\\
&\leq 2 \cdot \Pr\left[G_0^A \text{ sets } \mathsf{bad}\right] + 2 \cdot \Pr\left[G_1^A \Rightarrow 1\right] - 1\;. && (4)
\end{aligned}$$

To conclude the proof, we will design $A_{\mathbb{G}}$ and A_{SE} so that

$$\Pr\left[G_0^A \text{ sets } \mathsf{bad}\right] \leq \mathbf{Adv}_{\mathbb{G}}^{\text{gap-dh}}(A_{\mathbb{G}}) \tag{5}$$

$$2 \cdot \Pr\left[G_1^A \Rightarrow 1\right] - 1 \leq \mathbf{Adv}_{\mathsf{SE}}^{\text{ind-cca}}(A_{\mathsf{SE}}) . \tag{6}$$

Inequality (1) follows from (4), (5) and (6). We will now justify the above claims.

To justify (2), we claim that game G_0 perfects mimics the game defining the ind-cca-advantage of A. Game G_0 begins by picking the secret and public key for recipient 1, the challenge bit c, and an initial state (r, R) for the sender. For any public key X, where either $X = X_1$ or A queried $\textsc{MkBdRec}(X)$, the game maintains as $K[R, X]$ a k-bit key that plays the role of $H(R, X, X^r)$. This value is picked ahead of time as needed to answer queries to the $\textsc{Encrypt}$, LOR and $\textsc{Decrypt}$ oracles. Note that neither x_1 nor r are used in replying to these queries. In replying to random-oracle queries, the game "patches" H to return $K[R, X]$ to query R, X, X^r, and returns a random value if the query does not have this form. The patching code is broken into cases for $X = X_1$ and $X \neq X_1$, but since line 509 is included in G_0, the result is to always patch as we have just said. Game G_0 sets the flag bad when A makes the "crucial" random-oracle query R, X_1, X_1^r, the response to which should be the symmetric key $K[R, X_1]$ under which the challenge message is symmetrically encrypted. However, the flag does not affect the oracle responses. It also saves X_1^r as D^* in this case. Note that in responding to H queries, the value r is used, at line 502.

Equation (3) is obtained merely by adding and subtracting $\Pr[G_1^A \Rightarrow 1]$.

Games G_0, G_1 are identical-until-bad as defined in [11], meaning differ only in statements that follow the setting of the flag bad to true. The Fundamental Lemma of Game Playing [11] thus says that $\Pr[G_0^A \Rightarrow 1] - \Pr[G_1^A \Rightarrow 1] \leq \Pr[G_0^A \text{ sets } \mathsf{bad}]$. This justifies (4).

Adversary $A_{\mathbb{G}}$ has as input g and group elements that we (suggestively) denote as R, X_1. It also has access to the restricted DDH oracle $\textsc{ddh}_g(R, \cdot, \cdot)$. It begins by executing lines 003, 004 of **Initialize**, so that quantities $R, X_1, c, n, K[R, X_1]$ corresponding to the ones in G_0 are defined. (But $A_{\mathbb{G}}$ does not know r, x_1.) Now it runs A on inputs g, X_1, replying to A's oracle queries as per the procedures of G_0. We argue that it can do this. This is easy for $\textsc{MkBdRec}$, $\textsc{Encrypt}$, LOR and $\textsc{Decrypt}$ queries because the corresponding procedures have already been written to not use r or x_1 and can thus be executed as shown. But $A_{\mathbb{G}}$ can also execute the procedure to reply to H queries because it can call its $\textsc{ddh}_g(R, \cdot, \cdot)$ oracle on inputs X, D to implement the line 502 test as to whether $D = X^r$. When A halts, $A_{\mathbb{G}}$ ignores its output bit d. If the value D^* is defined (meaning if line 507 was executed) then $A_{\mathbb{G}}$ outputs D^*, and else it outputs some default value, like $1 \in \mathbb{G}$. Clearly if R, X_1 are random group elements then $A_{\mathbb{G}}$ perfectly mimics game G_0. The value D^* is defined iff bad is set to true, and, if defined, has value X_1^r. We have justified (5).

In Game G_1, the reply to query $H(R, X_1, X_1^r)$ will not be the "correct" value $K[R, X_1]$ but rather a random, unrelated value. We take advantage of this to design adversary A_{SE}. It begins by executing lines 001, 002 and 004 of **Initialize**. It has oracles that we suggestively denote by $\mathsf{SEnc}(K[R, X_1], \cdot)$, $\text{LOR}(K[R, X_1], \cdot, \cdot, c)$, $\mathsf{SDec}(K[R, X_1], \cdot)$, meaning we are identifying the key underlying these oracles with $K[R, X_1]$ and the challenge bit c with that of Game G_1, so that quantities $R, X_1, c, n, K[R, X_1]$ corresponding to the ones in G_1 are defined. (But $A_{\mathbb{G}}$ does not know $K[R, X_1]$ or c.) Now it runs A on inputs g, X_1, replying to A's oracle queries as per the procedures of G_0. We argue that it can do this. For $\textsc{MkBdRec}$, it can execute the procedure as shown. For $\textsc{Encrypt}$, it uses its $\mathsf{SEnc}(K[R, X_1], \cdot)$ oracle. For LOR, it uses its $\text{LOR}(K[R, X_1], \cdot, \cdot, c)$ oracle.

For Decrypt, to simulate line 404, it uses its $\mathsf{SDec}(K[R, X_1], \cdot)$ oracle if $S = R$ and otherwise runs the code shown in that line. It can execute G_1's procedure to reply to H queries as shown because it does have r and also because line 509 is omitted. When A halts with an output bit d, A_{SE} also outputs d and halts. If $K[R, X_1]$ is a random k-bit string and c is a random bit then A_{SE} has perfectly simulated G_1, and thus $\Pr[c = d]$, in the game defining the ind-cca-advantage of A_{SE}, equals $\Pr[G_1^A \Rightarrow 1]$. We have justified (6). ∎

5 The stateful KD scheme

This section describes our stateful version StKD of the Kurosawa-Desmedt [30] variant of the Cramer-Shoup [21] scheme.

Building blocks. We describe the building blocks used and assumptions made about them:

- A cyclic group $\mathbb{G}$ whose order is denoted m. We assume m is a prime. Let $\mathbb{G}^* = \mathbb{G} - \{1\}$ denote the set of generators of $\mathbb{G}$. We assume the DDH problem is hard in $\mathbb{G}$, captured by defining the ddh-advantage of an adversary $A_{\mathbb{G}}$ as
$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathbb{G}}(A_{\mathbb{G}}) = \Pr[\, A_{\mathbb{G}}(g_1, g_2, g_1^r, g_2^r) = 1 \,] - \Pr[\, A_{\mathbb{G}}(g_1, g_2, g_1^r, g_2^s) = 1 \,] \;,$$
where $g_1, g_2 \xleftarrow{\$} \mathbb{G}^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_m$.
- A symmetric encryption scheme $\mathsf{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ with keylength k. It is assumed IND-CPA secure [8], captured by defining the ind-cpa-advantage of an adversary A_{SE} as
$$\mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathsf{SE}}(A_{\mathsf{SE}}) = 2 \cdot \Pr\left[\, d = c \;:\; K \xleftarrow{\$} \{0,1\}^k \;;\; c \xleftarrow{\$} \{0,1\} \;;\; d \xleftarrow{\$} A_{\mathsf{SE}}^{\mathsf{SEnc}(K,\cdot),\mathrm{LOR}(K,\cdot,\cdot,c)} \,\right] - 1 \;,$$
where the LOR oracle is as in Section 4. We also assume integrity of ciphertexts (INT-CTXT) [27, 10], captured by defining the int-ctxt-advantage of an adversary A_{SE} as
$$\mathbf{Adv}^{\mathrm{int\text{-}ctxt}}_{\mathsf{SE}}(A_{\mathsf{SE}}) = \Pr\left[\, A_{\mathsf{SE}}^{\mathsf{SEnc}(K,\cdot),\mathsf{SDec}(K,\cdot)} \text{ forges} \;:\; K \xleftarrow{\$} \{0,1\}^k \,\right] \;,$$
where A_{SE} forges it it makes a query C to $\mathsf{SDec}(K, \cdot)$ such that the latter does not return $\perp$ but C was not previously returned by $\mathsf{SEnc}(K, \cdot)$. Note IND-CPA+INT-CTXT is (strictly) stronger than IND-CCA [10], but schemes with this property are easily obtained via the encrypt-then-mac construction [10].
- A family $H\colon\ \{0,1\}^k \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_m$ of hash functions in which $H(K, \cdot, \cdot)\colon\ \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_m$ for each k-bit key K. It is assumed universal one-way [31], also called TCR [12], captured by defining the tcr-advantage of an adversary A_H as
$$\mathbf{Adv}^{\mathrm{tcr}}_{H}(A_H) = \Pr[\, H(K, S_1, T_1) = H(K, S_2, T_2) \wedge (S_1, T_1) \neq (S_2, T_2) \,]$$
in the experiment where we first run A_H to get (S_1, T_1), only then pick key K at random, given K to A_H, and have it now provide (S_2, T_2) [31]. Note TCR is a weaker requirement than collision-reistance, so that, in particular, any practical collision-resistant function can be used.
- A key-derivation function $F\colon\ \mathbb{G} \to \{0,1\}^k$. We assume its output on a random input is computationally indistinguishable from a random k-bit string, captured by defining the kdf-advantage of an adversary A_F as
$$\mathbf{Adv}^{\mathrm{kdf}}_{F}(A_F) = \Pr[\, A_F(F(S)) = 1 \,] - \Pr[\, A_F(K) = 1 \,] \;,$$
where $S \xleftarrow{\$} \mathbb{G}$ and $K \xleftarrow{\$} \{0,1\}^k$. Such functions are easily built out of cryptographic hash functions.

The assumptions are the same as made in [25] for the KD scheme except that we need a randomized symmetric encryption scheme whereas a one-time scheme sufficed in [25]. The cost of upgrading the symmetric scheme is however negligible.

Scheme. The Setup algorithm of the *stateful KD scheme* $\mathsf{StKD} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{PKCk}, \mathsf{NwSt}, \mathsf{Enc}, \mathsf{Dec})$ returns as system parameters a pair g_1, g_2 of generators chosen at random from $\mathbb{G}^*$ and a random k-bit key K_H for H. A receiver's secret and public keys, as created by $\mathsf{KG}(g_1, g_2)$, are (x_1, x_2, y_1, y_2) and X, Y respectively, where x_1, x_2, y_1, y_2 are chosen at random from $\mathbb{Z}_m$ and $X = g_1^{x_1} g_2^{x_2}$ and $Y = g_1^{y_1} g_2^{y_2}$. $\mathsf{PKCk}((g_1, g_2, K_H), (X, Y))$ does nothing, simply returning 1, since the conditions put on keys by the KSK model already imply that any public key arising in the system will be a possible output of $\mathsf{KG}((g_1, g_2, K_H))$ so that, in particular it will consist of a pair of group elements. The new state algorithm $\mathsf{NwSt}((g_1, g_2, K_H))$ picks r at random from $\mathbb{Z}_m$ and returns $st = (r, R_1, R_2, \alpha)$ where $R_1 = g_1^r$, $R_2 = g_2^r$ and $\alpha = H(K_H, R_1, R_2)$. Given system parameters g_1, g_2, K_H, public key (X, Y) assumed to consist of a pair of group elements, message M and state $st = (r, R_1, R_2, \alpha)$, encryption algorithm Enc sets $Z = X^r Y^{r\alpha}$, $K = F(Z)$, computes the symmetric ciphertext $C_s \stackrel{\$}{\leftarrow} \mathsf{SEnc}(K, M)$, returns (R_1, R_2, C_s) as the ciphertext, and returns an unmodified state (r, R_1, R_2, α). (As with StDH, encryption does not change the state, and all the randomness for the encryption comes from the symmetric encryption scheme.) Given system parameters (g_1, g_2, K_H), secret key (x_1, x_2, y_1, y_2) and ciphertext $C = (R_1, R_2, C_s)$, decryption algorithm Dec returns $\perp$ if R_1 or R_2 are not in $\mathbb{G}$ and otherwise sets $\alpha = H(K_H, R_1, R_2)$, $Z = R_1^{x_1 + y_1\alpha} R_2^{x_2 + y_2\alpha}$ and $K = F(Z)$, and returns $\mathsf{SDec}(K, C_s)$ (which may be $\perp$).

Instantiations and cost. Since a multi-exponentiation $a, b, A, B \mapsto A^a B^b$ has essentially the same cost as an exponentiation, the cost for public-key operations is one exponentiation each for encryption and decryption, namely one-third and the same, respectively as for KD, and the same as for StDH. State initialization costs two exponentiations, but this is expected to be done rarely, and in the worst case that it is done before each encryption, the total cost only returns to that of KD. The symmetric primitives used can be instantiated in practice via block ciphers and cryptographic hash functions.

Variant. As with StDH, the StKD scheme has the nice feature that the symmetric key K computed by the encryption algorithm is a deterministic function of the system parameters, state and public key. Thus it could be computed only the first time the sender communicates with a particular receiver and then cached in the state, reducing the cost of further encryptions to this receiver to merely symmetric operations.

Security. While the proof of security of this scheme does not rely on the random oracle model, we unfortunately have to make the assumption that we are in the KSK model. Recall this captures the assumption that the CA issues a certificate for a public key only upon receiving a proof of knowledge of the corresponding secret key. In practice, this would likely be implemented by having receivers run proofs of knowledge of representations of X and Y with respect to g_1 and g_2 (see [20, 16]) as part of the certification protocol. When a sender encrypts under a public key, it suffices that it simply verifies the certificate, as usual. Note that some security practitioners have long advocated a "proof of possession of secret key" when certifying a public key. The current application and those in [6, 15] are some that appear to benefit from this.

Theorem 5.1 *Let $\mathbb{G}$ be a group of prime order m in which the DDH problem is hard. Let $\mathsf{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ be an IND-CPA+INT-CTXT secure symmetric encryption scheme with keylength k. Let $H\colon \{0,1\}^k \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_m$ be a family of TCR hash functions. Let $F\colon \mathbb{G} \to \{0,1\}^k$ be a secure key derivation function as defined above. Then the StKD scheme associated to these primitives is IND-CCA secure in the stateful KSK model.* ∎

Theorem 5.1 can be proved by slightly modifying the proof of security of the ordinary KD scheme given in [25]. We sketch this at a very high level, but the reader should be able to easily fill in the details by consulting [25].

PROOF SKETCH. Let A be an adversary attacking StKD in our stateful KSK model. Let $st = (r^*, R_1^*, R_2^*, \alpha^*)$ denote the sender's state, which is generated at the beginning of the game, and remains fixed throughout, and let X^*, Y^* denote the public-key of the honest receiver (namely, receiver 1), and $(x_1^*, x_2^*, y_1^*, y_2^*)$ its corresponding secret key. We also define $Z^* = g_1^{r^*} g_2^{r^* \alpha^*}$ and $K^* = F(Z^*)$. For a given ciphertext $C = (R_1, R_2, C_s)$, we call (R_1, R_2) the *preamble*, and C_s the *body*. We call (R_1^*, R_2^*) the *target preamble*. We say a preamble (R_1, R_2) is *valid* if $\log_{g_1}(R_1) = \log_{g_2}(R_2)$.

The first step is to observe that we may assume A makes no ENCRYPT queries except those of the form ENCRYPT$(1, \cdot)$. This is because when the adversary registers a public key (X, Y) for a "dishonest receiver" (namely one whose index is $i > 1$), he must, according to the KSK model, present a corresponding secret key (x_1, x_2, y_1, y_2) such that $X = g_1^{x_1} g_2^{x_2}$ and $Y = g_1^{y_1} g_2^{y_2}$. Because he knows this secret key, he can encrypt any message M he likes with respect to the public key (X, Y) (and the given state) by computing

$$Z \leftarrow (R_1^*)^{x_1 + y_1 \alpha^*} (R_2^*)^{x_2 + y_2 \alpha^*} \; ; \; K \leftarrow F(Z) \; ; \; C_s \stackrel{\$}{\leftarrow} \mathsf{SEnc}(K, M) \; .$$

This is where we make critical use of the fact that all users must register private keys (or at least present a proof of knowledge to the CA). Note that to make this work, the adversary may need to make a preliminary query of the form ENCRYPT$(1, \cdot)$, just to obtain the target preamble. With this simplification, note that the ENCRYPT and LOR oracles encrypt messages using the same symmetric key K^*.

We further assume the adversary does not submit to the DECRYPT oracle a ciphertext that was previously output by the LOR or ENCRYPT oracles. This is justified, since (a) by definition, the adversary is not allowed to get an output of LOR decrypted, and (b) the adversary already "knows" the decryption of any ciphertext output by ENCRYPT.

We now have an attack game which is much closer to the standard (non-stateful) IND-CCA attack game. The only difference is that the same state is used to encrypt several messages, instead of just one. In the proof of [25], only a single message was encrypted using a given state, and because of that, a weaker assumption on the symmetric encryption scheme, namely that it was IND-CPA+INT-CTXT in a single encryption attack, sufficed. Having strengthened the assumption to IND-CPA+INT-CTXT in a multiple encryption attack, it is really just an exercise to modify the proof in [25] appropriately.

We give an outline of this proof here. The proof is a sequence of games [36, 11], Game 0, Game 1, etc. Game 0 is the original attack game (with the above simplifications). We denote by S_i the event that A guesses the hidden bit in Game i. Our goal is to show that $\Pr[S_0] \approx 1/2$.

Game 1. We modify the decryption oracle so that it rejects all ciphertexts whose preambles do not match the target preamble, yet hash to the same value. The TCR assumption on H implies that $\Pr[S_1] \approx \Pr[S_0]$.

Game 2. We modify the way that the challenger computes Z^* (and hence K^*): as

$$Z^* \leftarrow (R_1^*)^{x_1^* + y_1^* \alpha} (R_2^*)^{x_2^* + y_2^* \alpha^*} .$$

This is purely a conceptual change, so $\Pr[S_2] = \Pr[S_1]$.

Game 3. We now replace the target preamble by a randomly chosen, invalid preamble. The DDH assumption implies that $\Pr[S_3] \approx \Pr[S_2]$.

Game 4. We now modify the decryption oracle, so that it rejects any ciphertext whose preamble does not match the target preamble, and is invalid (note that to implement this efficiently, we now assume the challenger knows $\log_{g_1}(g_2)$). Let E_4 be the event that some ciphertext is rejected using this new rejection rule that would not have been otherwise rejected. We have $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[E_4]$. We want to show that $\Pr[E_4] \approx 0$. To do this, we augment Game 4 slightly: we have the challenger choose at random one particular decryption query, called the *chosen decryption query*, and define E'_4 to be the event that that particular ciphertext was rejected using the new rule. If Q is a bound on the number of decryption queries, then $\Pr[E_4] \leq Q \Pr[E'_4]$, and so it will suffice to show that $\Pr[E'_4] \approx 0$. We will establish this later, in Game $6'$.

Game 5. We make two changes:

- First, we replace Z^* by a completely random group element. This is done in a consistent fashion, so that not only the Encrypt and LOR oracles use this value of Z^*, but so does the decryption oracle on ciphertexts whose preamble matches the target preamble.
- Second, in processing the chosen decryption query, if the preamble is different from the target preamble, does not hash to the same value as the target preamble, and is invalid, then either the symmetric encryption algorithm will reject the ciphertext, or if not, the new rejection rule introduced in Game 4 will reject it. For the purposes of determining if the new rejection rule is needed, a random group element $\tilde{Z}$ is chosen, and the body of the ciphertext is decrypted using the symmetric key $\tilde{K} = F(\tilde{Z})$: if this decryption does not result in $\perp$, we say "the new rejection rule applies"; in any case, the decryption oracle returns $\perp$ to the adversary.

Let E'_5 denote the event that the new rejection rule applies in Game 5. We have $\Pr[S_5] = \Pr[S_4]$ and $\Pr[E'_5] = \Pr[E'_4]$. This follows from the usual 4-wise independence argument, as in the original Cramer-Shoup analysis: the joint distribution of all the relevant random variables defining these events is identical.

The proof now forks in two different directions, making two different modifications to Game 5.

Game 6. Now we replace K^* be a random k-bit key. This is done consistently throughout the game: in the LOR oracle, the Encrypt oracle, and the Decrypt oracle (for those ciphertexts whose preamble matches the target preamble). Under the assumption that F is a good key derivation function, we have $\Pr[S_6] \approx \Pr[S_5]$. Moreover, note that K^* is essentially used in this game as a key in a standard symmetric chosen ciphertext attack, and under the security assumption for SE, it follows that $\Pr[S_6] \approx 1/2$.

Game 6′. We now modify Game 5 in a different way. Instead of computing $\tilde{K}$ as $F(\tilde{Z})$, we simply choose $\tilde{K}$ as a random k-bit key. Define $E'_{6'}$ to be the event that the new rejection rule applies in Game 6. Under the assumption that F is a good key derivation function, we have $\Pr[E'_{6'}] \approx \Pr[E'_5]$. Moreover, under the assumption that SE provides ciphertext integrity, we must have $\Pr[E'_{6'}] \approx 0$. It follows now that $\Pr[E'_4] = \Pr[E'_5] \approx \Pr[E'_{6'}] \approx 0$, and hence $\Pr[S_4] \approx \Pr[S_3]$.

Tracing through the logic of the above sequence of games, we see that $\Pr[S_0] \approx \Pr[S_6] \approx 1/2$, which proves the theorem.

Note that from the proof, we see that SE does not need to provide full ciphertext integrity. The properties needed are that it is IND-CCA secure, and that for any randomly chosen key, and any adversarially chosen ciphertext, the ciphertext is rejected with overwhelming probability (in this game, the adversary has no access to an encryption oracle).

References

[1] M. Abe, R. Gennaro, K. Kurosawa and V. Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryption and a New Analysis of Kurosawa Desmedt KEM. *Advances in Cryptology – EUROCRYPT '05*, Lecture Notes in Computer Science Vol. 3494, R. Cramer ed., Springer-Verlag, 2005.

[2] M. Abdalla, M. Bellare and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. *Topics in Cryptology – CT-RSA '01*, Lecture Notes in Computer Science Vol. 2020, D. Naccache ed., Springer-Verlag, 2001.

[3] J. An. Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses. Cryptology ePrint Archive: Report 2001/079.

[4] J. An, Y. Dodis and T. Rabin. On the Security of Joint Signature and Encryption. *Advances in Cryptology – EUROCRYPT '02*, Lecture Notes in Computer Science Vol. 2332, L. Knudsen ed., Springer-Verlag, 2002.

[5] M. Bellare, A. Boldyreva and A. Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. *Advances in Cryptology – EUROCRYPT '04*, Lecture Notes in Computer Science Vol. 3027, C. Cachin and J. Camenisch ed., Springer-Verlag, 2004.

[6] M. Bellare, A. Boldyreva and J. Staddon. Multi-Recipient Encryption Schemes: Security Notions and Randomness Re-Use. *Public-Key Cryptography '03*, Lecture Notes in Computer Science Vol. 2567, Y. Desmdedt ed., Springer-Verlag, 2003.

[7] M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[8] M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. *Proceedings 38th Symposium on Foundations of Computer Science*, IEEE, 1997.

[9] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[10] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000.

[11] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. *Advances in Cryptology – EUROCRYPT '06*, Lecture Notes in Computer Science Vol. 4004, S. Vaudenay ed., Springer-Verlag, 2006.

[12] M. Bellare and P. Rogaway. Collision-Resistant Hashing: Towards Making UOWHFs Practical. *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.

[13] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.

[14] D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity Based Encryption. *Topics in Cryptology – CT-RSA '05*, Lecture Notes in Computer Science Vol. 3376, A. Menezes ed., Springer-Verlag, 2005.

[15] A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap Diffie-Hellman Group Signature Scheme. *Public-Key Cryptography '03*, Lecture Notes in Computer Science Vol. 2567, Y. Desmdedt ed., Springer-Verlag, 2003.

[16] S. Brands. Untraceable off-line cash in wallets with observers. *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[17] M. Burmester and Y. Desmedt. Remarks on soundness of proofs. *Electronics Letters*, 25(22), 1509–1511, 1989.

[18] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *Proceedings 30th Annual Symposium on the Theory of Computing*, ACM, 1998.

[19] R. Canetti, S. Halevi and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *Advances in Cryptology – EUROCRYPT '04*, Lecture Notes in Computer Science Vol. 3027, C. Cachin and J. Camenisch ed., Springer-Verlag, 2004.

[20] D. Chaum, J. Evertse and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. *Advances in Cryptology – EUROCRYPT '87*, Lecture Notes in Computer Science Vol. 304, D. Chaum ed., Springer-Verlag, 1987.
[21] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1), 167–226, 2003.
[22] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal of Computing*, 30(2), 391–437, 2000.
[23] A. Desai. New Paradigms for Constructing Symmetric Encryption Schemes Secure against Chosen Ciphertext Attack. *Advances in Cryptology – CRYPTO '00*, Lecture Notes in Computer Science Vol. 1880, M. Bellare ed., Springer-Verlag, 2000.
[24] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), 469–472, 1985.
[25] R. Gennaro and V. Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive: Report 2004/194.
[26] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, 28, 270–299, 1984.
[27] J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. *Fast Software Encryption '00*, Lecture Notes in Computer Science Vol. 1978, B. Schneier ed., Springer-Verlag, 2000.
[28] E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. *Theory of Cryptography – TCC '06*, Lecture Notes in Computer Science Vol. 3876, S. Halevi and T. Rabin ed., Springer-Verlag, 2006.
[29] K. Kurosawa. Multi-Recipient Public-Key Encryption with Shortened Ciphertext. *Public-Key Cryptography '02*, Lecture Notes in Computer Science Vol. 2274, D. Naccache and P. Paillier ed., Springer-Verlag, 2002.
[30] K. Kurosawa and Y. Desmedt. A New Paradigm of Hybrid Encryption Scheme. *Advances in Cryptology – CRYPTO '04*, Lecture Notes in Computer Science Vol. 3152, M. Franklin ed., Springer-Verlag, 2004.
[31] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *Proceedings 21st Annual Symposium on the Theory of Computing*, ACM, 1989.
[32] National Institute of Standards and Technology (NIST). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Document SP 800-38B, May 2005.
[33] T. Okamoto and D. Pointcheval. The Gap Problems: A New Class of Problems for the Security of Cryptographic Schemes. *Public-Key Cryptography '01*, Lecture Notes in Computer Science Vol. 1992, K. Kim ed., Springer-Verlag, 2001.
[34] C. Rackoff and D. Simon. Non-interactive zero knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.
[35] P. Rogaway, M. Bellare and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information System Security*, 6(3), 365–403, 2003.
[36] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive: Report 2004/332.
[37] V. Shoup. Why chosen ciphertext security matters. IBM Research Report RZ 3076, November 1998.
[38] Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.

A Extended security model and equivalence with basic one

In Section 3 we presented a fairly simple basic security model that we have used for the results and proofs in the paper. Here we show that security in this model implies security in a richer, extended model.

To define IND-CCA security of stateful public-key encryption scheme $\mathsf{StPE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{PKCk}, \mathsf{NwSt}, \mathsf{Enc}, \mathsf{Dec})$ in the extended model we consider a game played with an adversary A_e. (The "e" stands for "extended.") We first describe this for the USK model and then discuss what changes for the KSK model. The game begins with the initializations

$$sp \xleftarrow{\$} \mathsf{Setup}\ ;\ c \xleftarrow{\$} \{0,1\}\ ;\ n \leftarrow 0\ ;\ \mathrm{GR}, \mathrm{GS} \leftarrow \emptyset\ .$$

A_e is then given input sp. The following shows what types of oracle queries A_e can make, what action the game takes in each case, and what response the game returns to A_e:

Oracle query	**Action taken by game**	**Response**
$\textsc{MkGdSen}()$	$s \leftarrow s+1\ ;\ st_s \leftarrow \mathsf{NwSt}(sp)\ ;\ \mathrm{GS} \leftarrow \mathrm{GS} \cup \{s\}$	s
$\textsc{Reset}(j)$	$st_j \xleftarrow{\$} \mathsf{NwSt}(sp)\ ;\ \mathrm{GS} \leftarrow \mathrm{GS} \cup \{j\}$	1
$\textsc{MkGdRec}()$	$n \leftarrow n+1\ ;\ (pk_n, sk_n) \xleftarrow{\$} \mathsf{KG}(sp)\ ;\ \mathrm{GR} \leftarrow \mathrm{GR} \cup \{n\}$	n
$\textsc{MkBdRec}(pk)$	If $\mathsf{PKCk}(sp, pk) = 1$ then $n \leftarrow n+1\ ;\ pk_n \leftarrow pk$	n
$\mathrm{LOR}(i, j, M_0, M_1)$	$(C, st_j) \xleftarrow{\$} \mathsf{Enc}(sp, pk_i, M_c, st_j)$	C
$\textsc{Encrypt}(i, j, M)$	$(C, st_j) \xleftarrow{\$} \mathsf{Enc}(sp, pk_i, M, st_j)$	C
$\textsc{RevealSt}(j)$	$\mathrm{GS} \leftarrow \mathrm{GS} - \{j\}$	st_j
$\textsc{CrptRec}(i)$	$\mathrm{GR} \leftarrow \mathrm{GR} - \{i\}$	sk_i
$\textsc{Decrypt}(i, C)$	$M \leftarrow \mathsf{Dec}(sk_i, C)$	M

A_e can call $\textsc{MkGdSen}()$ to create a new sender whose index is stored in the set GS of good senders. A_e can call $\textsc{Reset}(j)$ to reset the state of sender j, with the restriction that $1 \leq j \leq s$. The lifetime of sender j is viewed as divided into periods, the p-th period ending with the p-th $\textsc{Reset}(j)$ query that A_e makes ($p \geq 1$). A_e can call $\textsc{MkGdRec}()$ to create a new receiver whose index is added to the set GR of good receivers and whose keys are chosen honestly by the game, with the public key returned to A_e. A_e can call $\textsc{MkBdRec}(pk)$, and if pk passes the public-key verification check then the game creates a receiver with public key pk. A_e can call $\mathrm{LOR}(i, j, M_0, M_1)$ to have sender j encrypt challenge message M_c under its current state and the public key of receiver i, and return the ciphertext to A_e, with the restriction that $j \in \mathrm{GS}$, $i \in \mathrm{GR}$ and $|M_0| = |M_1|$. Unlike in the basic model, we do not restrict A_e to just one LOR query. A_e can call $\textsc{Encrypt}(i, j, M)$ to have sender j encrypt M under its current state and the public key of receiver i, and return the ciphertext to A_e, with the restriction that $j \in \mathrm{GS}$ and $1 \leq i \leq n$. A_e can call $\textsc{RevealSt}(j)$ to reveal the current state of sender j, modeling a compromise of j's private information, with the following restrictions. First, $j \in \mathrm{GS}$. Second, in any period in the lifetime of j in which a $\mathrm{LOR}(\cdot, j, \cdot, \cdot)$ query has been made, no $\textsc{RevealSt}(j)$ query is allowed. This captures the goal that a sender can recover from compromise by resetting its state: as long as the current state has not been exposed, its communications should remain private. A_e can call $\textsc{CrptRec}(i)$ to expose the secret key of receiver i, with the restriction that $i \in \mathrm{GR}$ and no $\mathrm{LOR}(i, \cdot, \cdot, \cdot)$ query is ever made. A_e can call $\textsc{Decrypt}(i, C)$ to obtain the decryption of C under the secret key of receiver i, with the restriction that $i \in \mathrm{GR}$ and C has not been returned in response to a previous LOR query. This models a chosen-ciphertext attack. At the end of its execution, A must output a bit. Denoting it by d, the ind-cca-e-advantage of A is

$$\mathbf{Adv}^{\text{ind-cca-e}}_{\mathsf{StPE}}(A) = 2 \cdot \Pr\left[d = c\right] - 1\ .$$

For the KSK model, the difference is like in the basic case. Namely $\textsc{MkBdRec}$ takes input not just pk but also sk and A_e is restricted to queries where pk, sk is a possible output of $\mathsf{KG}(sp)$. The action taken by the game is simply $n \leftarrow n+1\ ;\ pk_n \leftarrow pk$, and the response returned is n.

Theorem A.1 *Let* StPE *be a stateful public-key encryption scheme. Let* A_e *be a USK (resp. KSK) adversary against* StPE *in the extended model, making at most* $q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9$ *queries, respectively, to its oracles in the order they are listed in the table above, and having running time* *at most t. Then there is a USK (resp. KSK) adversary A against* StPE *in the basic model such that*

$$\mathbf{Adv}^{\text{ind-cca-e}}_{\mathsf{StPE}}(A_e) \leq q_5(q_1 q_2)q_3 \cdot \mathbf{Adv}^{\text{ind-cca}}_{\mathsf{StPE}}(A) \ .$$

Furthermore, A makes $q_4 + q_3 - 1, q_6 + q_5 - 1, q_9$ queries to its MkBdRec, Encrypt, Decrypt *oracles respectively. Its running time is that of A_e plus the time for a number of invocations of the algorithms of the scheme that is linear in the sum of the number of queries.* ▌

Proof Sketch: We will prove this for the USK case. The KSK case is similar. We will evolve A_e into a sequence of simpler adversaries $A_e^1, A_e^2, \ldots$, still in the rich model, until we get one that is easily realized in the basic model. The table below summarizes the query counts made by these adversaries and the last line shows the factor loss in advantage involved in the moving to this adversary from the previous one:

Oracle	A_e	A_e^1	A_e^2	A_e^3
MkGdSen()	q_1	q_1	1	1
Reset(j)	q_2	q_2	0	0
MkGdRec()	q_3	q_3	q_3	1
MkBdRec(pk)	q_4	q_4	q_4	q_4+q_3-1
LOR(i, j, M_0, M_1)	q_5	1	1	1
Encrypt(i, j, M)	q_6	q_6+q_5-1	q_6+q_5-1	q_6+q_5-1
RevealSt(j)	q_7	q_7	0	0
CrptRec(i)	q_8	q_8	q_8	0
Decrypt(i, C)	q_9	q_9	q_9	q_9
Factor loss:		q_5	$q_1 q_2$	q_3

First use a hybrid argument to get an adversary A_e^1 that makes only one LOR query. To simulate the other LOR queries (due to the hybrid, it knows which of the two challenge messages is being encrypted here) it will use the Encrypt oracle, so the number of queries to this goes up as shown. Adversary A_e^2 guesses the sender j and the period t in its lifetime such that the single LOR query is from j and made in period t. It simulates all other senders for all their time periods, and also simulates all other time periods of j by starting from a fresh state each time. It needs no RevealSt queries since it simulates answers for all but the crucial one, and that one is illegal if the guess is right. A_e^3 guesses the identity i of the honest receiver in the (single) LOR query, and simulates all MkGdRec queries other than the one creating i via MkBdRec, itself picking the public and secret keys, so that it can answer CrptRec queries for them. (Such a query is illegal to the guessed receiver if the guess is right.) From the profile of A_e^3 we see that it is effectively an adversary in the basic model. ▌