Intrusion-Resilient Secure Channels

Full Version[⋆]

Gene Itkis[⋆⋆], Robert McNerney Jr.[⋆⋆⋆], Scott Russell[†]

Boston University Computer Science Dept.
111 Cummington St.
Boston, MA 02215, USA
`{itkis,robmcn,srussell}@bu.edu`

Abstract. We propose a new secure communication primitive called an *Intrusion-Resilient Channel (IRC)* that limits the damage resulting from key exposures and facilitates recovery. We define security against passive but mobile and highly adaptive adversaries capable of exposing even expired past secrets. We describe an intuitive channel construction using (as a black box) existing public key cryptosystems. The simplicity of the construction belies the technical challenges in its security proof.
Additionally, we outline a general strategy for proving enhanced security for two-party protocols when an IRC is employed to secure all communication. Specifically, given a protocol proved secure against adversaries with restricted access to protocol messages, we show how the use of an IRC allows some of these adversary restrictions to be lifted. Once again, proving the efficacy of our intuitive approach turns out to be non-trivial. We demonstrate the strategy by showing that the intrusion-resilient signature scheme of [IR02] can be made secure against adversaries that expose even expired secrets.

1 Introduction

1.1 Motivation and Contributions

Background. One of the most basic problems in cryptography is that of secure communication between two parties, call them Alice and Bob. Typically, Alice and Bob ensure confidentiality and integrity of their conversation using a primitive called a *secure channel* that encrypts and authenticates their messages using coordinated secret key(s). As long as only Alice and Bob know these secret keys, the channel guarantees their messages remain secure. Once another party learns these keys, confidentiality of all messages sent using these keys is lost.

Due to its importance, the secure channel problem has been widely researched, leading to numerous results, many of which have been implemented in practice. Widely deployed cryptographic secure channels include SSH, TLS/SSL, IPSEC, various VPNs, etc. See [MVV97] for a survey of the rich history of this problem. Secure channels also form an important building block in protocols for performing more complex tasks ranging from on-line auctions to secure multi-party computation. Much of the relevant work to date has focused on initializing the channel, leading to the study of (authenticated) key exchange: from the earliest ideas of Diffie and Hellman [DH76] to the more refined and formalized extensions of [BR93,BCK98,Sho99,CK01], to name just a few.

[⋆] An extended abstract of this work will appear in the proceedings of Applied Cryptography and Network Security 2005

[⋆⋆] Supported in part by National Science Foundation grant No. CCR-0311485.
[⋆⋆⋆] Supported by U.S. Dept. of Education GAANN award #P200A040102
[†] Supported in part by first author's NSF grant and by a National Physical Science Consortium Fellowship with stipend support from the National Security Agency.

In contrast, our main goal is to limit the loss of confidentiality due to exposures of Alice and/or Bob's secret keys by an adversarial entity and to facilitate channel recovery. For signatures and encryption these goals are achieved in *intrusion-resilient* schemes [IR02,DFK$^+$03] by combining key evolution ideas from forward-secure schemes [And97,BM99] and secret-sharing from proactive schemes [OY91,HJJ$^+$97]. Key evolution changes the secret key over time in such a way that prior keys cannot be computed from the current key, thus limiting the loss of confidentiality to messages sent with current and future keys. Secret-sharing distributes shares of the secret among multiple parties who proactively refresh the sharing by exchanging refresh messages. Consequently, to learn the secret the adversary must expose shares of multiple parties at the same time.

Channel Definition and Construction. In Section 2 we formulate a new secure two-party communication primitive called an *Intrusion-Resilient Channel (IRC)* that uses familiar key-evolution and proactive techniques to limit propagation of exposures both forward and backward in time and facilitates restoration of confidential communication. In Section 3 we describe $g\mathcal{IRC}$, our intrusion-resilient channel construction based on any semantically secure public key encryption scheme with sufficiently large domain[1]. $g\mathcal{IRC}$ uses public key encryption in a straightforward way to secure the two-way channel traffic. Either party may proactively refresh the channel by generating new key pairs and sending the appropriate public and private keys to the other party. Despite the simplicity of the construction, its proof of security requires a surprising amount of work, since refresh messages include new private keys for the recipient.

Channel Applications. In Section 4 we explore the message hiding capabilities of general two-party protocols augmented with an intrusion-resilient channel. Specifically, in Section 4.3 we outline a general strategy for proving the increased security of channel-augmented protocols. To demonstrate the strategy, we prove the intrusion-resilient signature scheme of [IR02] secure against adversaries with greater temporal adaptivity than was previously proved. We believe the strategy will work for other two-party protocols as well, such as intrusion-resilient encryption [DFK$^+$03] and proactive two-party signatures [NKDM03].

A key component of our strategy is Theorem 3 which is developed in Section 4.2. Theorem 3 in turn relies on several useful adversary restrictions and a modified notion of protocol simulation developed in Section 4.1. A key feature of our protocol simulators is the ability to measure their success, i.e. indistinguishability, with respect to specified *subsets* of the views they generate. We believe this type of simulator may be interesting in its own right and applicable in other contexts.

1.2 Adversary Models and Assumptions

Protocol Adversaries. General protocol adversaries may be *static* or *mobile*. Parties corrupted by static adversaries remain corrupted until the protocol terminates. Mobile adversaries, introduced in [OY91], capture the idea that corrupted parties may be able to detect intrusions and execute a recovery mechanism, effectively removing the adversary and restoring the party to an uncorrupted state. When considering mobile adversaries, it is useful to talk about *secret exposures* rather than full corruption in order to specify to a finer degree the knowledge the adversary learns as the result of various actions.

[1] The public key encryption scheme must also be key-indistinguishable as defined in [BBDP01] for $g\mathcal{IRC}$ to be used as a secure sub-protocol (see Section 4).

Adaptive adversaries determine their next action based on the results of all previous exposures and other queries, whereas *non-adaptive* adversaries must specify their entire activity schedule at the time of protocol initialization.

Active adversaries may modify messages sent by uncorrupted parties and exert complete control over corrupted parties, for example by arbitrarily modifying their internal state and messages and causing them to deviate from the protocol. In contrast, *passive* adversaries are only allowed to observe the internal state of corrupted parties, and may read protocol messages without modifying them.

Our Model and Assumptions. Adversaries in this work are assumed to be adaptive and mobile (even able to ask about "expired" information) but passive. Parties are not assumed to possess long term secrets which cannot be exposed, but all parties are assumed to have access to independent sources of *private randomness*, that is, they can privately generate truly random bits (even *after* exposure). To simplify the proofs communication is assumed to be *synchronous*, that is, all messages are delivered instantaneously and reliably. In practice only an ordered, reliable datagram delivery mechanism similar to TCP is needed.

Although results exist for active, adaptive and mobile adversaries (see Section 1.3), restricting to passive adversaries allows us to provide the desired resiliency and recovery capability via a simple, generic construction. In particular, we avoid altogether the need for authentication which is difficult to maintain against active adversaries. Despite its simplicity, significant technical challenges are encountered in proving the message-privacy properties of our construction.

1.3 Comparison with Previous Work

Adaptive Mobile Adversaries. In [CHH00] Canetti et al. show how to restore authenticity to a party in the presence of active, adaptive and mobile adversaries. This is accomplished by using a "proactive distributed signature scheme" which they realize in a setting without authenticated communication. Other proactive threshold[2] schemes include [FMY99], built on the non-adaptive scheme of [FGMY97], and [HJJ$^+$97], based on [HJKY95] (see below). All such threshold schemes require an honest majority of communicating parties, and thus are not applicable to the case of two-party protocols.

[CK02] recasts the classical notion of secure channels into the powerful universally composable (UC) model [Can01] by defining key exchange and secure channel functionalities in the presence of *active*, adaptive and mobile adversaries. They realize relaxed versions of the intuitive functionalities by using a modification to the UC model called "non-information" oracles. These oracles help bridge the differences between indistinguishability and simulation based security. However, their exposure model assumes that the active adversary does not learn the long-term authentication secret of a party, allowing them to avoid the complexities of impersonation attacks considered in [CHH00].

In contrast, to avoid authentication altogether we assume an adversary that is passive but who learns the *entire* state of a party upon exposure. This allows us to focus on damage containment and recovery, the hallmarks of existing intrusion-resilient schemes, at the expense of the secure composability of UC model. Indeed, the security proofs of the intrusion-resilient signature and encryption schemes of [IR02,DFK$^+$03], two potential applications for our IRC, are not in the UC

[2] A *threshold* scheme remains secure provided the number of *simultaneously* compromised parties never exceeds the given threshold.

model. An interesting line of future research would be to combine the intrusion-resilient and UC models together.

Non-Adaptive Mobile Adversaries. Secret sharing in the presence of non-adaptive mobile adversaries is handled in [HJKY95]. To aid recovery, all parties broadcast channel refresh messages according to a proactive "private key renewal protocol". In contrast, in this paper a refresh consists of a single party sending a single message. This was done to provide consistency with the existing intrusion-resilient signature and public key encryptions schemes of [IR02] and [DFK$^+$03]. The proof techniques in this paper should suffice to prove the key renewal protocol of [HJKY95] intrusion-resilient (and "spliceable"; see Section 2.3) against *adaptive* adversaries. We emphasize that this key renewal protocol was not articulated as a stand-alone primitive.

2 Definitions

2.1 Functional Definition: Two-party Key-Evolving Channel

Let $u \in \{0,1\}$ and $\bar{u} = 1-u$ denote the identities of the two communicating parties, and let $\tau > 0$ be a natural number denoting a time period. The channel is fully bi-directional, so that in all that follows u and $\bar{u}$ may be interchanged. Let $SK_\tau^{[u]}$ denote the secret key for party u in period τ. Similarly, $R_\tau^{[u]}$ denotes the key refresh message *received* by party u for period τ which is generated and sent by party $\bar{u}$. Party u combines refresh information in $R_\tau^{[u]}$ with its current key $SK_\tau^{[u]}$ to obtain its new key $SK_{\tau+1}^{[u]}$ for the next period $(\tau+1)$ and then deletes the expired $SK_\tau^{[u]}$, thereby completing the *refresh protocol*. The current period τ thus gives the number refreshes executed since channel initiation. It is assumed that τ is explicitly part of both $SK_\tau^{[u]}$ and $R_\tau^{[u]}$. For convenience let the array U contain the identity of the refresh message receiver for each period, that is $U[\tau] \in \{0,1\}$ denotes the receiver for period τ. Note that the channel communicates two types of messages[3]: data and refresh.

Definition 1. *A* two-party key-evolving channel $\mathcal{KEC} = (\mathit{InitKeys}, \mathit{EncM}, \mathit{DecM}, \mathit{GenR}, \mathit{ProcR}; \underline{\mathit{SendMesg}}, \underline{\mathit{RefKeys}})$ *is a quintuple of algorithms and pair of protocols as described in Figure 1.*

2.2 Security Definition: Channel Intrusion-Resilience

A $\mathcal{KEC}$ adversary $\mathcal{A}$ is modeled as probabilistic polynomial-time (PPT) Turing machine that conducts an adaptive chosen-ciphertext (CCA2) attack[4] against $\mathcal{KEC}$ with the aid of the oracles in O_t according to the security experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{channel\text{–}ind\text{–}b}$ detailed in Figure 2. Let Q denote the sequence of all oracle queries made by $\mathcal{A}$ during a run of experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{channel\text{–}ind\text{–}b}(k)$ in Figure 2. In addition to being exposed *directly* by querying *Okey*, keys may be exposed *indirectly* due to the combined results of specific *Okey* and *Oref* queries. This leads to a notion of key exposure analogous to that in [IR02].

[3] In practice, a mechanism for distinguishing the two types like a distinguished 1-bit flag in the (unencrypted) message header is needed. We omit this detail henceforth.

[4] Appropriately restricting the set O_t yields definitions of intrusion-resilient security against ciphertext-only and adaptive chosen-message attacks. Additional definitions can be created by specifying the appropriate oracles for these notions.

Fig. 1. Two-party Key-evolving Channel $\mathcal{KEC}$ Algorithms and Protocols (underlined)

$\mathcal{KEC}.\mathit{InitKeys}(1^k[,\ldots]) \to \langle SK_0^{[0]}, SK_0^{[1]}\rangle$**:** channel initialization algorithm

- **In:** Security parameter k (in unary)
- **Out:** Key pair $\langle SK_0^{[0]}, SK_0^{[1]}\rangle$ to be used by the channel endpoints

$\mathcal{KEC}.\mathit{EncM}_{SK_\tau^{[u]}}(m) \to c$**:** message encryption algorithm

- **In:** Message m to be sent from party u to party $\bar{u}$, current period secret key $SK_\tau^{[u]}$ of the sending party
- **Out:** Ciphertext c

$\mathcal{KEC}.\mathit{DecM}_{SK_\tau^{[\bar{u}]}}(c) \to m'$**:** message decryption algorithm

- **In:** Ciphertext c (received by party $\bar{u}$ from party u), current period secret key $SK_\tau^{[\bar{u}]}$ of the receiving party
- **Out:** Message m', where $\forall m\ \mathit{DecM}_{SK_\tau^{[\bar{u}]}}(\mathit{EncM}_{SK_\tau^{[u]}}(m)) = m$

$\mathcal{KEC}.\mathit{GenR}(SK_\tau^{[u]}) \to \langle SK_{\tau+1}^{[u]}, R_\tau^{[\bar{u}]}\rangle$**:** key refresh generation algorithm

- **In:** Current period secret key $SK_\tau^{[u]}$
- **Out:** Refreshed secret key $SK_{\tau+1}^{[u]}$ and key refresh message $R_\tau^{[\bar{u}]}$ for party $\bar{u}$.
- Party u sends $R_\tau^{[\bar{u}]}$ to $\bar{u}$, securely deletes $SK_\tau^{[u]}$ and $R_\tau^{[\bar{u}]}$, and sets $\tau = \tau + 1$.

$\mathcal{KEC}.\mathit{ProcR}_{SK_\tau^{[\bar{u}]}}(R_\tau^{[\bar{u}]}) \to SK_{\tau+1}^{[\bar{u}]}$**:** key refresh processing algorithm

- **In:** Secret key $SK_\tau^{[\bar{u}]}$, key refresh $R_\tau^{[\bar{u}]}$ (received by party $\bar{u}$ from party u)
- **Out:** Refreshed secret key $SK_{\tau+1}^{[\bar{u}]}$. Party $\bar{u}$ securely deletes $SK_\tau^{[\bar{u}]}$ and $R_\tau^{[\bar{u}]}$, and sets $\tau = \tau + 1$.

$\mathcal{KEC}.\underline{\mathit{SendMesg}}$**:** Protocol for party u to securely send message m to party $\bar{u}$.

1. Party u calls $\mathit{EncM}_{SK_\tau^{[u]}}(m)$ and sends output c to $\bar{u}$.
2. The other party $\bar{u}$ on receipt of data message c from u, calls $\mathit{DecM}_{SK_\tau^{[\bar{u}]}}(c)$ to retrieve the sent plaintext data message m.

$\mathcal{KEC}.\underline{\mathit{RefKeys}}$**:** Protocol for refreshing the secret keys of both parties.

1. Party u initiates refresh by calling $\mathit{GenR}(SK_\tau^{[u]})$ which outputs $(SK_{\tau+1}^{[u]}, R_\tau^{[\bar{u}]})$.
2. Party u sends refresh message $R_\tau^{[\bar{u}]}$ to $\bar{u}$ (GenR securely deletes $SK_\tau^{[u]}$ and $R_\tau^{[\bar{u}]}$, and increments τ by 1).
3. The other party $\bar{u}$ on receipt of $R_\tau^{[\bar{u}]}$ completes the refresh cycle by calling $\mathit{ProcR}_{SK_\tau^{[\bar{u}]}}(R_\tau^{[\bar{u}]})$ which returns $SK_{\tau+1}^{[\bar{u}]}$ (ProcR securely deletes $SK_\tau^{[\bar{u}]}$ and $R_\tau^{[\bar{u}]}$, and increments τ by 1).

Definition 2 (Q-exposure). *For a sequence of adversary oracle queries Q and time period τ, we say that the key $SK_\tau^{[u]}$ is Q-*exposed[5]*:*

[directly] *if $(u, \tau) \in Q$; or*

[via evolution] *if $SK_{\tau-1}^{[u]}$ is Q-exposed* **and** *$U[\tau-1] = u$* **and** *$\tau - 1 \in Q$*
(Given $SK_{\tau-1}^{[u]}$ and $R_{\tau-1}^{[u]}$, $\mathcal{A}$ can compute $SK_\tau^{[u]}$ as $\mathcal{KEC}.\mathit{ProcR}_{SK_{\tau-1}^{[u]}}(R_{\tau-1}^{[u]})$).

Channel Compromise and Recovery. Whereas key exposure models the "internal" side of a security breach, compromise refers to its "external" aspect. We say that party u is (Q, τ)-compromised if $SK_\tau^{[u]}$ is Q-exposed. As long as party u is Q-exposed, the adversary can read all messages directed toward u by simply decrypting them using $SK^{[u]}$. If both sides of the channel are (Q, τ)-compromised for the same period τ then the channel is totally compromised for period τ, i.e. neither refresh nor data messages can be securely exchanged.

An important feature of intrusion-resilient channels, due the passive adversary assumption, is their ability to recover from even a total compromise. Party u recovers from a compromise as soon as it sends a refresh message, even if this message is observed and read. Party $\bar{u}$ if also compromised, can likewise recover by initiating its own refresh after receiving party u's. Thus,

[5] Alternate models allowing additional exposure types may have more efficient constructions with weaker security are not explored in this paper.

channel recovery is achieved whenever both parties send refreshes without any intervening direct exposures. Moreover, if the adversary ever fails to intercept a refresh, then both the refresh sender and receiver simultaneously recover from compromise.

Security Experiment. As in the standard security definitions for public key cryptosystems (see Appendix A.1), $\mathcal{A}$ is divided into probing and distinguishing subcomponents: $\mathcal{A}_{probe}$ and $\mathcal{A}_{dist}$. Note that $\mathcal{A}_{probe}$ maintains state during the experiment, and passes its final state to $\mathcal{A}_{dist}$. $\mathcal{A}$ specifies the timing and direction of each channel refresh. $\mathcal{A}$ can obtain encryptions and decryptions on adaptively chosen messages for the party, i.e. direction, and time period of its choice via the oracle in the set O_t. Eventually, $\mathcal{A}$ requests a challenge ciphertext for a message pair, party (direction), and time period of its choice, and attempts to distinguish. It is required that $\mathcal{A}$ not query $Odec$ on the challenge ciphertext or Q-expose the corresponding decryption secret.

Definition 3 (Intrusion-Resilience).
Let $\mathcal{KEC} = (\mathit{InitKeys}, \mathit{EncM}, \mathit{DecM}, \mathit{GenR}, \mathit{ProcR}; \underline{\mathit{SendMesg}}, \underline{\mathit{RefKeys}})$ *be a two-party key evolving channel scheme with security parameter k, and let* $\mathcal{A} = (\mathcal{A}_{probe}, \mathcal{A}_{dist})$ *be an adversary. The (CCA)* advantage *of* $\mathcal{A}$ *against the channel* $\mathcal{KEC}$ *is defined as:*
$\mathbf{Adv}^{channel\text{–}ind}_{\mathcal{KEC},\mathcal{A}}(k) \overset{def}{=} \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}1}_{\mathcal{KEC},\mathcal{A}}(k)=1] - \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}0}_{\mathcal{KEC},\mathcal{A}}(k)=1]$.
$\mathcal{KEC}$ *is (CCA)* intrusion-resilient *if* $\mathbf{Adv}^{channel\text{–}ind}_{\mathcal{KEC},\mathcal{A}}(\cdot)$ *is negl. for all PPT* $\mathcal{A}$.

2.3 Channel Spliceability

As we will see when we explore the message-hiding capabilities of two-party protocols augmented with an intrusion-resilient channel in Section 4, our results rely on an additional hiding property of our channel. Namely, data messages sent over the channel should leak no information about the specific key under which they were encrypted. In the context of public key cryptosystems this property is called *key indistinguishability* and was introduced by Bellare, et al. in [BBDP01]. For our channels we formulate an appropriate version of this property called *spliceability*, which we define in detail in Appendix A.2.

3 A Generic Intrusion-Resilient Channel Construction

This section describes $g\mathcal{IRC}$, our intrusion-resilient channel, which can be constructed using any semantically secure, public key cryptosystem[6] $\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$. $g\mathcal{IRC}$ consists of two bi-directional sub-channels, one to secure data messages and the other to secure refresh messages. Each party holds a pair of public keys for sending and a pair of secret keys for receiving on each sub-channel. Either party may proactively refresh the channel by generating a complete set of new keys for both sides of the channel. The generating party retains its own new sub-channel keys. It sends the complimentary keys to the other party in a refresh message encrypted under the appropriate (expiring) refresh sub-channel key.

Despite the simplicity of $g\mathcal{IRC}$, its security proof (see proof sketch below) is complicated by the fact that refresh messages include the recipient's new secret keys. A detailed description of $g\mathcal{IRC}$ appears in Figure 3.

[6] See Appendix A.1 for the definition of a public key cryptosystem. Any $\mathcal{E}$ with domain sufficiently large enough to accommodate the messages described in Figure 3 can be used. If $\mathcal{E}$ is also key-private then the resulting $g\mathcal{IRC}$ is spliceable.

Fig. 2. Intrusion Resilience Security Experiment

```
Oracle Set[a] $O_t$
  $Okey_t(u,\tau)$: key exposure oracle
    on input party identifier $u$ and period $\tau \le t$ returns $SK_\tau^{[u]}$.
  $Oenc_t(u,\tau,m)$: message encryption oracle
    on input party identifier $u$, period $\tau \le t$, and message $m$ returns $EncM_{SK_\tau^{[u]}}(m)$.
  $Odec_t(\bar{u},\tau,c)$: message decryption oracle
    on input party identifier $\bar{u}$, period $\tau \le t$, and ciphertext $c$ returns $DecM_{SK_\tau^{[\bar{u}]}}(c)$.
  $Oref_t(\tau)$: key refresh message oracle
    on input period $\tau \le t$ returns $R_\tau^{[U[\tau]]}$.
```

```
Experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{channel\text{-}ind\text{-}b}(k)$
INIT:
  $t \leftarrow 0$; $q \leftarrow$ ref(0,1); $c \leftarrow \bot$;
  $\langle SK_t^{[0]}, SK_t^{[1]} \rangle \leftarrow InitKeys(1^k)$;
PROBE:
  while $q$ in (ref($w,\bar{w}$),"test") do
    $q \leftarrow \mathcal{A}_{probe}^{O_t}(1^k)$
    CASE $q$
      ref($w,\bar{w}$): % send refresh from party $w$ to party $\bar{w}$
        $\langle SK_{t+1}^{[w]}, R_t^{[\bar{w}]} \rangle \leftarrow GenR(SK_t^{[w]})$ % party $w$ generates new key and refresh
        $SK_{t+1}^{[\bar{w}]} \leftarrow ProcR_{SK_t^{[\bar{w}]}}(R_t^{[\bar{w}]})$ % party $\bar{w}$ receives refresh and updates its key
        $U[t] \leftarrow \bar{w}$ % party $\bar{w}$ is recorded as the receiver for period $t$
        $t \leftarrow t+1$ % time period incremented
      "test": % output challenge
        if $c \ne \bot$ then break % only one challenge allowed[b]
        $(u,\bar{u},\tau,m_0,m_1) \leftarrow \mathcal{A}_{probe}^{O_t}(1^k)$ % request period $\tau$ challenge, $(u,\bar{u})$ direction[c]
        $c \leftarrow Oenc(m_b,u,\tau)$ % encrypt challenge with $SK_\tau^{[u]}$; decryptable with $SK_\tau^{[\bar{u}]}$
        $\mathcal{A}_{probe}^{O_t}(1^k,c)$ % $\mathcal{A}_{probe}$ receives challenge $c$
DIST:
  $s \leftarrow \mathcal{A}_{probe}^{O_t}(1^k)$ % $\mathcal{A}_{probe}$ outputs state $s$
  $\hat{b} \leftarrow \mathcal{A}_{dist}^{O_t}(1^k,s)$ % $\mathcal{A}_{dist}$ outputs distinguishing guess $\hat{b}$ for challenge bit $b$
  if $SK_\tau^{[\bar{u}]}$ not $Q$-exposed % decryption key not exposed (receiver not compromised)
    and $(\mathcal{A}_{probe}, \mathcal{A}_{dist})$ did not query $Odec$ with $(\bar{u},\tau,c)$ % challenge not decrypted
    then return $\hat{b}$
  else return $\bot$
```

[a] O_t denotes the set of oracles to which $\mathcal{A}$ has access to during period t. All oracles take as input τ, the period in which the adversary is interested. Queries about the future ($\tau > t$) are prohibited, and result in the oracle returning $\bot$. Without loss of generality, we assume this never happens.

[b] Limiting $\mathcal{A}$ to a single challenge is not an essential restriction. Given an adversary $\mathcal{A}'$ which is allowed $poly(k)$ challenges, one can easily construct an $\mathcal{A}$ with advantage one over $poly(k)$ the advantage of $\mathcal{A}'$.

[c] We require that u, w, and $\hat{b}$ all be from $\{0,1\}$. If $\tau > t$, then $Oenc_t$ returns $\bot$. The adversarially chosen messages m_0 and m_1 must of course be in the domain of $EncM_{SK_\tau^{[u]}}(\cdot)$ and be of equal length, i.e. $|m_0| = |m_1|$.

Construction Details. During a given time period τ, channel party u maintains the $g\mathcal{IRC}$ key $SK_\tau^{[u]} = \langle sk_\tau^{[u]}, pk_\tau^{[\bar{u}]} \rangle$. The key $sk_\tau^{[u]}$ is used to decrypt incoming messages, while $pk_\tau^{[\bar{u}]}$ is used to encrypt the outgoing ones. Furthermore, $sk_\tau^{[u]} = (sk_\tau^{[u|d\rangle}, sk_\tau^{[u|\rho\rangle})$ and $pk_\tau^{[\bar{u}]} = (pk_\tau^{[\bar{u}|d\rangle}, pk_\tau^{[\bar{u}|\rho\rangle})$. The $\mathcal{S}$ keys $sk^{[\cdot|d\rangle}, pk^{[\cdot|d\rangle}$ are used to secure the channel data traffic, while $sk^{[\cdot|\rho\rangle}, pk^{[\cdot|\rho\rangle}$ secure the channel maintenance traffic (i.e., the refresh messages).

Party u initiates a refresh during period τ by invoking *GenR*. This in turn calls *InitKeys*, which uses $\mathcal{G}$ to assemble new composite keys $SK_{\tau+1}^{[u]}, SK_{\tau+1}^{[\bar{u}]}$ for both sides of the channel. Party u encrypts the new key $SK_{\tau+1}^{[\bar{u}]}$ for party $\bar{u}$ as $\mathcal{E}_{SK_\tau^{[u]}.pk_\tau^{[\bar{u}|\rho\rangle}}(SK_{\tau+1}^{[\bar{u}]})$ and sends this to $\bar{u}$. As soon as a party obtains its new key, its expiring key is securely deleted. Recall that in our model[7], $\bar{u}$'s secret $SK_{\tau+1}^{[\bar{u}]}$ cannot be exposed by exposing u. Clearly, some synchronization issues must be addressed to ensure that the refresh messages are not sent simultaneously from different endpoints. This should be taken care of in the *RefKeys* protocol (namely, ensuring that at any time only a single *RefKeys* protocol is running).

Fig. 3. Generic Intrusion-Resilient Channel $g\mathcal{IRC}$ Algorithms

```
$g\mathcal{IRC}.GenKeys(1^k)$
    $(sk^{[0|d\rangle}, pk^{[0|d\rangle}) \leftarrow \mathcal{G}(1^k)$
    $(sk^{[0|\rho\rangle}, pk^{[0|\rho\rangle}) \leftarrow \mathcal{G}(1^k)$
    $(sk^{[1|d\rangle}, pk^{[1|d\rangle}) \leftarrow \mathcal{G}(1^k)$
    $(sk^{[1|\rho\rangle}, pk^{[1|\rho\rangle}) \leftarrow \mathcal{G}(1^k)$
    $SK^{[0]} \leftarrow \langle (sk^{[0|d\rangle}, sk^{[0|\rho\rangle}), (pk^{[1|d\rangle}, pk^{[1|\rho\rangle}) \rangle$
    $SK^{[1]} \leftarrow \langle (sk^{[1|d\rangle}, sk^{[1|\rho\rangle}), (pk^{[0|d\rangle}, pk^{[0|\rho\rangle}) \rangle$
    return $(SK^{[0]}, SK^{[1]})$
```

```
$g\mathcal{IRC}.GenR(SK_\tau^{[u]})$
    $(SK_{\tau+1}^{[0]}, SK_{\tau+1}^{[1]}) \leftarrow GenKeys(1^k)$
    $R_\tau^{[\bar{u}]} \leftarrow \mathcal{E}_{SK_\tau^{[u]}.pk^{[\bar{u}|\rho\rangle}}(SK_{\tau+1}^{[\bar{u}]})$
    securely delete $SK_\tau^{[u]}$ and $SK_{\tau+1}^{[\bar{u}]}$
    return $(SK_{\tau+1}^{[u]}, R_\tau^{[\bar{u}]})$
    $\tau \leftarrow \tau + 1$
```

```
$g\mathcal{IRC}.InitKeys(1^k)$
    $(SK_0^{[0]}, SK_0^{[1]}) \leftarrow GenKeys(1^k)$
    return $(SK_0^{[0]}, SK_0^{[1]})$
```

```
$g\mathcal{IRC}.ProcR(R_\tau^{[\bar{u}]}, SK_\tau^{[\bar{u}]})$
    $SK_{\tau+1}^{[\bar{u}]} \leftarrow \mathcal{D}_{SK_\tau^{[\bar{u}]}.sk^{[\bar{u}|\rho\rangle}}(R_\tau^{[\bar{u}]})$
    securely delete $SK_\tau^{[\bar{u}]}$
    return $SK_{\tau+1}^{[\bar{u}]}$
    $\tau \leftarrow \tau + 1$
```

```
$g\mathcal{IRC}.EncM(m, SK_\tau^{[u]})$
    return $c \leftarrow \mathcal{E}_{SK_\tau^{[u]}.pk^{[\bar{u}|d\rangle}}(m)$
```

```
$g\mathcal{IRC}.DecM(c, SK_\tau^{[\bar{u}]})$
    return $m \leftarrow \mathcal{D}_{SK_\tau^{[\bar{u}]}.sk^{[\bar{u}|d\rangle}}(c)$
```

Theorem 1 ($g\mathcal{IRC}$ is Intrusion-Resilient). *If $\mathcal{S}$ is a CCA semantically-secure public-key encryption scheme, $g\mathcal{IRC}$ is an intrusion-resilient channel.*

More precisely, given any $g\mathcal{IRC}$-adversary $\mathcal{A}$, one can construct a triple of $\mathcal{S}$-adversaries $\langle E_1, E_2, E_3 \rangle$ such that for each value of the security parameter k there exists some $i \in \{1,2,3\}$ such that, $\mathbf{Adv}_{\mathcal{S},E_i}^{ind\text{–}cca}(k) \geq \mathbf{Adv}_{g\mathcal{IRC},\mathcal{A}}^{channel\text{–}ind}(k)/(4q_{refr}{}^2 \cdot$
$(3q_{refr} - 2))$, where q_{refr} is an upper bound on the number of refreshes requested by the adversary in experiment $\mathbf{Exp}_{g\mathcal{IRC},\mathcal{A}}^{channel\text{–}ind\text{–}b}(k)$ (see Figure 2 in Section 2.2). The running times of the E_i are essentially the same as for $\mathcal{A}$.

[7] In more complicated adversary exposure models where $SK_{\tau+1}^{[\bar{u}]}$ could be exposed by corrupting u and stealing the key before it is sent it would be useful to consider slightly more complex and less generic mechanisms utilizing proactive encryption schemes. Such a model will be explored in future work.

Proof (Sketch). (Complete proof available in Appendix C). There are two cases to consider: the ciphertext challenge and refresh message beginning the challenge period specified by the adversary have either the *same* or *opposite* direction. If the direction is the same, a straightforward reduction to the semantic security of $\mathcal{S}$ is used. In the opposite case, a two step reduction is used. First the channel security is reduced to the security of a chain of $\mathcal{S}$ instances where the secret key of the next instance is encrypted under the public key of the current instance and given to the adversary. The security of this $\mathcal{S}$ chain is then reduced to the semantic security of a single $\mathcal{S}$ instance.

Theorem 2 ($g\mathcal{IRC}$ is Spliceable). *If $\mathcal{S}$ is a CCA semantically-secure public-key encryption scheme and $\mathcal{S}$ is CCA key-indistinguishable (IK-CCA) in the sense of [BBDP01], then $g\mathcal{IRC}$ is spliceable.*

Proof. [Sketch](Complete proof available in Appendix D). Follows by formulating a chain of intermediate definitions connecting key-indistinguishability to channel spliceability.

4 More Secure Two-Party Protocols via Intrusion Resilient Channels

In this section we explore the message-hiding capabilities of general two-party protocols when augmented with an intrusion-resilient channel. Suppose a two-party protocol $\mathcal{P}$ is secure against all adversaries who can only view a protocol message by exposing the receiver during the same period in which it was sent. Any messages for which this is not the case are called "forbidden" and are never seen by the adversary. One would expect $\mathcal{P}'$, an IRC-augmented version of $\mathcal{P}$, to be secure against adversaries who can view *any* protocol message, even forbidden ones.

However, this seems difficult to prove in the *adaptive* adversary setting considered here, since the set of messages which are forbidden depends dynamically on the adversary's query sequence. Indeed, a straightforward security reduction is frustrated by the fact that a simulator does not know ahead of time which messages will ultimately be forbidden. Consequently, at the time of a message query the simulator does not know whether or not it should substitute "garbage" in place of the true protocol message. Thus, even though a secure channel hides the contents of all forbidden messages, in general a simulator will not be able to capitalize on this fact.

To help avoid this dilemma, commonly known as the "selective-decryption problem", in Section 4.1 we add an extra restriction to our definition of forbidden messages (Definition 4, condition 2), require the order of adversary exposures to be "refresh-receiver-biased" (Definition 6), and introduce a new type of protocol simulator (Definition 8). The success of these simulators is measured in terms of indistinguishability on a subset of their output (Definition 7).

These restrictions suffice to prove a result about simulators rather than adversaries (Theorem 3 in Section 4.2). However, in Section 4.3 we outline a general proof strategy for applying Theorem 3 to prove that if the two-party protocol $\mathcal{P}$ is secure against refresh-receiver-biased adversaries who cannot expose any forbidden messages, then the channel-augmented version $\mathcal{P}'$ is secure against any refresh-receiver-biased adversary who *can* expose forbidden messages. To demonstrate the strategy, we prove the intrusion-resilient signature scheme of [IR02] secure against adversaries with greater temporal adaptivity than in the proof of [IR02].

4.1 Protocol Security Model

In what follows, all protocol parties are modeled as interactive Turing machines (ITMs) [Gol01]. Recall that ITMs have a random tape, a work tape, local input and output tapes, and a pair of communication tapes for incoming and outgoing protocol messages. Let $\mathcal{P}$ be any two-party protocol with security parameter(s)[8] κ, initial input $\gamma_{\mathcal{P}}$, and randomness $r_{\mathcal{P}}$. Let $\mathcal{A}$ be an adversary for $\mathcal{P}$ with input $\gamma_{\mathcal{A}}$ and randomness $r_{\mathcal{A}}$. $\mathcal{A}$ also gets security parameter κ (in unary) as an input.

Correspondingly, let $\mathcal{A}'$ denote an adversary for $\mathcal{P}'$, a protocol identical to $\mathcal{P}$ but augmented so that all inter-party communication is protected using $\mathcal{IRC}$, a spliceable, intrusion-resilient channel. $\mathcal{P}'$ is constructed from $\mathcal{P}$ as follows. At the beginning of protocol $\mathcal{P}'$, in addition to the usual protocol secrets the parties also receive their respective initial keys for $\mathcal{IRC}$. Next, whenever in $\mathcal{P}$ party u would send message m, $\mathcal{P}'$ instead sends an encryption of m under party u's current channel key[9]. The sender u then executes a channel refresh and sends $\bar{u}$ the resulting channel refresh message[10].

Protocol Adversary Interface. In attacking $\mathcal{P}$, the adversary $\mathcal{A}$ may interact with the protocol $\mathcal{P}$ via its user interface, which includes the ability to execute cryptographic routines and specify protocol control commands to effect the evolving state of the protocol. Additionally, $\mathcal{A}$ has additional "attack" capabilities, such as the ability to expose secret keys and protocol messages or exert greater influence over the evolution $\mathcal{P}$. We use O_t to denote the set of oracles[11] which mediate these interactions between $\mathcal{A}$ and $\mathcal{P}$. Similarly, O'_t denotes the oracle set for $\mathcal{A}'$ attacking $\mathcal{P}'$.

Adversary Query Classes. Let Q denote the sequence of all oracle queries and control commands made by $\mathcal{A}$ while interacting with a particular execution of $\mathcal{P}$, and let $\mathcal{Q}$ denote a set of adversary query sequences, i.e. a set of Qs. Let Q' and $\mathcal{Q}'$ be similarly defined for adversary $\mathcal{A}'$ and protocol $\mathcal{P}'$.

Definition 4. *A message m sent in period τ using a key-evolving channel $\mathcal{KEC}$ is* **Q-forbidden** *if: 1) the receiver of m is not Q-exposed for period τ, and 2) the sender of m sent the previous (period $\tau - 1$) refresh.*

Let $Q_{FM} \subset Q$ denote the set of all Q-forbidden messages. Generalizing, let $\mathcal{Q} \setminus \mathcal{Q}_{FM}$ denote the set of query sequences derived from $\mathcal{Q}$ by removing from each $Q \in \mathcal{Q}$ its corresponding Q_{FM}.

Definition 5. *An interactive Turing machine I is* **$\mathcal{Q}$-restricted** *if its sequence of outgoing messages Q is always in $\mathcal{Q}$.*

Definition 6. *An adversary exposure sequence Q is* **refresh-receiver-biased** *if for all periods τ in which both ends of the channel are Q-exposed, the $\tau - 1$ refresh receiver is Q-exposed before the sender. A class $\mathcal{Q}$ of exposure query sequences is refresh-receiver-biased if $\forall Q \in \mathcal{Q}$, Q is refresh-receiver-biased. An adversary $\mathcal{A}$ is refresh-receiver-biased if $\mathcal{A}$ is $\mathcal{Q}$-restricted and $\mathcal{Q}$ is refresh-receiver-biased.*

[8] If more than one, let κ denote a vector of security parameters.

[9] Note that the granularity of channel time periods may differ from that of the protocol. Also, if necessary, m can be fragmented into multiple sub-messages prior to encryption.

[10] Other compositions are possible. For example, after every message sent, both parties could execute refreshes one right after the other, etc.

[11] While queries pertaining to past and current time periods are permissible, queries about the future and non-existent messages are prohibited (return value $\perp$). WLOG, we assume this never happens. In cases where $\mathcal{A}$ is given protocol control capability beyond that available to an honest $\mathcal{P}$ user, the particulars of this interface are handled by the security experiment in which $\mathcal{A}$ is participating. Note when the adversary is interacting with a simulator for $\mathcal{P}$, all oracle queries are handled by the simulator.

Indistinguishability with Respect to Subsets of Distributions. In the following experiment X and Y denote distributions over length n sequences of bit-strings. For a sequence of bit-strings x, $x[i]$ denotes the ith bit-string. Let $J \subseteq \{1, \ldots, n\}$ be an index subset. $\mathcal{D}^x$ denotes oracle access to x by $\mathcal{D}$ on a per bit-string basis.

```
Experiment $\mathbf{Exp}^{J\text{–}ind\text{–}b}_{X/Y,\mathcal{D}}(\kappa)$
if $b = 0$ then $x \leftarrow X$
if $b = 1$ then $x \leftarrow Y$
$\hat{b} \leftarrow \mathcal{D}^x$
if $\mathcal{D}$ queried for bit-string $x[j]$ for $j \notin J$ then output $\perp$     % $\mathcal{D}$ may only view bit-stings indexed by $J$
else output $\hat{b}$
```

Definition 7 (**J-indistinguishable distributions**). *Let J, X, and Y be as in the above experiment, and let $\mathcal{D}$ be a distinguishing adversary. We define the* J-**distinguishing advantage** *of $\mathcal{D}$ against distributions X and Y as:*
$\mathbf{Adv}^{J\text{–}ind}_{X/Y,\mathcal{D}}(\kappa) \overset{def}{=} \Pr[\mathbf{Exp}^{J\text{–}ind\text{–}0}_{X/Y,\mathcal{D}}(\kappa)=1] - \Pr[\mathbf{Exp}^{J\text{–}ind\text{–}1}_{X/Y,\mathcal{D}}(\kappa)=1]$. *We say the two distributions X and Y are* J-**indistinguishable**, *denoted $X \approx_J Y$, if $\mathbf{Adv}^{J\text{–}ind}_{X/Y,\mathcal{D}}(\cdot)$ is negligible for all PPT $\mathcal{D}$. More generally, we say that two distributions X and Y are* $\mathcal{J}$-**indistinguishable**, *denoted $X \approx_{\mathcal{J}} Y$, if $\forall J \in \mathcal{J}$, $\mathbf{Adv}^{J\text{–}ind}_{X/Y,\mathcal{D}}(\cdot)$ is negligible for all PPT $\mathcal{D}$.*

Let $\mathsf{VIEW}^{\mathcal{P},\mathcal{A}^{O_t}}_{\mathcal{A}}(\gamma_{\mathcal{P}}, r_{\mathcal{P}} | \gamma_{\mathcal{A}}, r_{\mathcal{A}})$ denote adversary $\mathcal{A}$'s view of its interaction with an instance of protocol $\mathcal{P}$ via the oracle set O_t. Recall that $\gamma_{\mathcal{A}}$ and $r_{\mathcal{A}}$ ($\gamma_{\mathcal{P}}$ and $r_{\mathcal{P}}$) denote the input and randomness for $\mathcal{A}$ ($\mathcal{P}$), and the security parameter κ is an implicit input to all parties. This view consists of the sequence of all queries made by $\mathcal{A}$ along with the corresponding responses received. Similarly, let $\mathsf{VIEW}^{\mathcal{P}',\mathcal{A}'^{O'_t}}_{\mathcal{A}'}(\gamma_{\mathcal{P}'}, r_{\mathcal{P}'} | \gamma_{\mathcal{A}'}, r_{\mathcal{A}'})$ denote the view of $\mathcal{A}'$ interacting via the oracle set O'_t with an instance of $\mathcal{P}'$. In the definition below $J_{Q'} \subseteq Q'$ denotes a unique (non-empty) subset of query sequence Q' used to qualify the indistinguishability of the views. $\mathcal{J}$ denotes the collection of all such $J_{Q'}$ for all $Q' \in \mathcal{Q}'$.

Definition 8. *An ITM Sim is a* $\mathcal{Q}'$**-answering,** $\mathcal{Q}$**-restricted,** ε**-good** $\mathcal{J}$**-simulator** *for protocol $\mathcal{P}$ with security parameter κ if*

1. *Sim is $\mathcal{Q}$-restricted,*
2. *Sim interactively generates query sequence $Q \in \mathcal{Q}$ to ask $\mathcal{P}$ in response to interactive query sequence Q'*
3. *$\exists$ an efficiently computable predicate $Sim.OK : \mathcal{Q}' \times \mathcal{Q} \to \{0, 1\}$*
4. *$\forall Q' \in \mathcal{Q}'$, $\Pr[Sim.OK(Q', Q) = 1] \geq \varepsilon(\kappa)$ taken over the random choices of Sim*
5. *$\forall$ ITM $\mathcal{A}$ asking $Q' \in \mathcal{Q}'$, if Q satisfies $Sim.OK(Q', Q) = 1$, then*
   $\mathsf{VIEW}^{\mathcal{P},\mathcal{A}^{O_t}}_{\mathcal{A}}(\gamma_{\mathcal{P}}, r_{\mathcal{P}} | \gamma_{\mathcal{A}}, r_{\mathcal{A}}) \approx_{J_{Q'}} \mathsf{VIEW}^{Sim,\mathcal{A}^{O_t}}_{\mathcal{A}}(\gamma_{Sim}, r_{Sim} | \gamma_{\mathcal{A}}, r_{\mathcal{A}})$.

Intuitively, Sim simulates answers to any query sequence $Q' \in \mathcal{Q}'$ by asking a transformed query sequence $Q \in \mathcal{Q}$ of $\mathcal{P}$. Sim succeeds with probability at least ε in producing a view which is indistinguishable when judged with respect to the smaller query set $J_{Q'} \subseteq Q'$.

4.2 Intrusion-Resilient and Spliceable Channels Hide Forbidden Messages

In what follows, let Sim' denote the ITM obtained by augmenting Sim with $\mathcal{IRC}$ in the same way that $\mathcal{P}'$ is obtained from $\mathcal{P}$. That is, Sim' augments the view produced by Sim to include $\mathcal{IRC}$ encrypted messages and keys. Sim' passes query sequence Q' to Sim which in turn interactively generates query sequence Q to answer the Q' queries. Note that $Sim'.OK \overset{def}{=} Sim.OK$[12].

Theorem 3. *Let $\mathcal{P}'$ be instantiated with an intrusion-resilient and spliceable channel. If Q' is refresh-receiver-biased and Sim is a $\mathcal{Q}'$-answering,*
$\mathcal{Q}$-restricted, ε-good $\mathcal{Q}' \setminus \mathcal{Q}'_{FM}$-simulator for $\mathcal{P}$, then Sim' is a $\mathcal{Q}'$-answering, $\mathcal{Q}$-restricted, ε-good $\mathcal{Q}'$-simulator for $\mathcal{P}'$.

Theorem 3 says that composing an intrusion-resilient and spliceable channel with a simulator Sim which answers query sequences $Q' \in \mathcal{Q}'$ but may only be indistinguishable when forbidden messages are excluded, i.e. for $Q' \setminus Q'_{FM}$, yields a simulator Sim' answering the same set of query sequences $\mathcal{Q}'$ and which is indistinguishable even on forbidden messages. For forbidden message queries Sim need only generate responses of the correct length. Note that Sim may have to deal with arbitrary simultaneous compromise of both ends of the channel, and thus a total compromise of $\mathcal{P}'$ in the two-party case.

Proof. By definition Sim' satisfies the first four simulator properties. The indistinguishability of views (Property 5) follows directly from Lemma 1, Lemma 2, and Lemma 3 (see below).

In the following lemmas $\mathcal{P}'$, $\mathcal{Q}'$, Sim, and Sim' are as in Theorem 3, and $\mathcal{A}'$ is a $\mathcal{P}'$ adversary. $O'_{S,t}$ denotes the oracle set O'_t modified such that the key exposure oracle returns random keys in place of the original data sub-channel keys for forbidden message senders.

Lemma 1. *For all $\mathcal{Q}'$-restricted $\mathcal{A}'$,*
$\mathsf{VIEW}_{\mathcal{A}'}^{\mathcal{P}',\mathcal{A}'^{O'_t}}(\gamma_{\mathcal{P}'}, r_{\mathcal{P}'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'}) \approx_{\mathcal{Q}'} \mathsf{VIEW}_{\mathcal{A}'}^{\mathcal{P}',\mathcal{A}'^{O'_{S,t}}}(\gamma_{\mathcal{P}'}, r_{\mathcal{P}'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'})$.

Proof. Follows from assumption that $\mathcal{IRC}$ is spliceable.

Lemma 2. *For all $\mathcal{Q}'$-restricted $\mathcal{A}'$,*
$\mathsf{VIEW}_{\mathcal{A}'}^{Sim',\mathcal{A}'^{O'_t}}(\gamma_{Sim'}, r_{Sim'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'}) \approx_{\mathcal{Q}'} \mathsf{VIEW}_{\mathcal{A}'}^{Sim',\mathcal{A}'^{O'_{S,t}}}(\gamma_{Sim'}, r_{Sim'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'})$.

Proof. Nearly identical to that for Lemma 1.

Lemma 3. *Conditioned on the event that $Sim'.OK(Q', Q) = 1$,*
$\mathsf{VIEW}_{\mathcal{A}'}^{\mathcal{P}',\mathcal{A}'^{O'_{S,t}}}(\gamma_{\mathcal{P}'}, r_{\mathcal{P}'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'}) \approx_{\mathcal{Q}'} \mathsf{VIEW}_{\mathcal{A}'}^{Sim',\mathcal{A}'^{O'_{S,t}}}(\gamma_{Sim'}, r_{Sim'}|\gamma_{\mathcal{A}'}, r_{\mathcal{A}'})$.

Proof. (Full proof in Appendix E).

[12] For all pairs (Q', Q), we should always have $Sim.OK(Q', Q) = 1 \Rightarrow Sim'.OK(Q', Q) = 1$, since if Sim does not need to abort with unencrypted simulated messages, Sim' with encrypted simulated messages should not have to either. The reverse implication may not always hold, but by assuming it we only underestimate the success (non-abort) probability of Sim'.

4.3 Improved Two-Party Protocol Security

A Specific Example. Let SiBIR2 denote the channel-augmented version of SiBIR1, the Signer-Base Intrusion-Resilient signature scheme of [IR02], in which all message traffic is secured via some spliceable, intrusion-resilient channel $\mathcal{IRC}$, as was described for general two-party protocols in the beginning of Section 4.1. Recall in SiBIR1 the long-term signing secrets are shared between Base and Signer modules (the two parties), whereas the complete signing secret for the current time period is always held by the Signer.

Let $\mathcal{Q}'$ be any query class consisting of refresh-receiver-biased query sequences against SiBIR2 such that for each query sequence in $\mathcal{Q}'$ there is at least one time period for which the Signer keys are not Q-exposed[13]. Provided these two conditions are satisfied, a sequence $Q' \in \mathcal{Q}'$ may contain both Base and Signer key exposures and message exposures in any order.

Theorem 4. SiBIR2 *is secure against all $\mathcal{Q}'$-restricted adversaries $\mathcal{A}'$, in particular any time-adaptive adversary restricted to refresh-receiver-biased query sequences.*

Proof (Sketch). (Proof available in Appendix F)

1. Select a suitably restricted SiBIR1 adversary query class $\mathcal{Q}$.
2. Prove SiBIR1 secure against all $\mathcal{Q}$-restricted adversaries.
3. Exhibit a $\mathcal{Q}'$-answering, $\mathcal{Q}$-restricted, ε-good $\mathcal{Q}' \setminus \mathcal{Q}'_{FM}$ simulator for SiBIR1.
4. Apply Theorem 3 to get a simulator which converts any SiBIR2 adversary into a SiBIR1 adversary.
5. Observe that the loss in advantage is just $\varepsilon = 1/T$, where T is the number of time periods.

General Strategy. We believe that the strategy outlined in the proof sketch of Theorem 4 for the specific case of intrusion-resilient signatures can also be used to prove the security of other channel-augmented two-party cryptographic protocols, including intrusion-resilient encryption [DFK$^+$03] and proactive two-party signatures [NKDM03]. To do so, first decide on a query class $\mathcal{Q}'$. Then follow the steps outlined above, replacing SiBIR1 and SiBIR2 with $\mathcal{P}$ and $\mathcal{P}'$ respectively. Note that the challenge lies in selecting appropriate adversary query classes $\mathcal{Q}$ and $\mathcal{Q}'$ such that Steps 2 and 3 can be accomplished.

References

[And97] Ross Anderson. Invited lecture. In *Fourth ACM Conference on Computer and Communication Security* [CCS97]. (see also [And02]).

[And02] Ross Anderson. Two remarks on public key cryptology. Technical Report UCAM-CL-TR-549, University of Cambridge, Computer Laboratory, December 2002. http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-549.pdf.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9–13, 2001: proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–??, New York, NY, USA, 2001. Springer-Verlag Inc.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In ACM, editor, *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, Texas, 23–26 May 1998.

[13] This is reasonable since no successful forger can expose the Signer keys for the forgery period.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 23–27 August 1998.

[BH93] Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT ' 92*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1993.

[BHNS99] Boaz Barak, Amir Herzberg, Dalit Naor, and Eldad Shai. The proactive security toolkit and applications. In Gene Tsudik, editor, *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 18–27, Singapore, November 1999. ACM Press.

[BM99] Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 15–19 August 1999. Revised version is available from `http://www.cs.ucsd.edu/~mihir/`.

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 22–26 August 1993.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, New York, October 2001. IEEE.

[CCS97] *Fourth ACM Conference on Computer and Communication Security*. ACM, April 1–4 1997.

[CHH00] Ran Canetti, Shai Halevi, and Amir Herzberg. Maintaining authenticated communication in the presence of break-ins. *Journal of Cryptology*, 13(1):61–105, January 2000.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 451–472. Springer-Verlag, 6–10 May 2001.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science, pages 337–351. Springer-Verlag, 28 April–2 May 2002.

[Des97] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proc. 1st International Information Security Workshop*, pages 158–173, 1997.

[DFK$^+$03] Y. Dodis, M. Franklin, J. Katz, A. Miyaji, and M. Yung. Intrusion-resilient public-key encryption. In *Progress in Cryptology — CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, 2003.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[FGMY97] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In *Proc. 17th International Advances in Cryptology Conference – CRYPTO '97*, pages 440–454, 1997.

[FMY99] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive rsa. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *Advances in Cryptology—ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*, pages 180–194, Singapore, 14–18 November 1999. Springer-Verlag.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[HJJ$^+$97] Amir Herzberg, Markus Jakobsson, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *Fourth ACM Conference on Computer and Communication Security* [CCS97], pages 100–110.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology—CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer-Verlag, 27–31 August 1995.

[IR02] Gene Itkis and Leonid Reyzin. SiBIR: Intrusion-resilient signatures, or towards obsoletion of certificate revocation. In Yung [Yun02]. Available from `http://eprint.iacr.org/2002/054/`.

[Itk02] Gene Itkis. Intrusion-resilient signatures: Generic constructions, or defeating strong adversary with minimal assumptions. In *Third Conference on Security in Communication Networks (SCN'02)*, volume 2576 of *Lecture Notes in Computer Science*. Springer-Verlag, September 12–13 2002.

[Itk03] Gene Itkis. Cryptographic tamper evidence. In *10th ACM Conference on Computer and Communication Security*. ACM, October 27–30 2003. Also avaliable from `http://www.cs.bu.edu/~itkis/papers/`.

[JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 221–242. Springer-Verlag, 14–18 May 2000.

[MVV97] A. J. (Alfred J.) Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography.* The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[NKDM03] A. Nicolosi, M. Krohn, Y. Dodis, and D. Mazières. Proactive two-party signatures for user authentication. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '03)*, 2003.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks. In *10-th Annual ACM Symp. on Principles of Distributed Computing*, pages 51–59, 1991.

[Sho99] Victor Shoup. On formal models for secure key exchange. Research Report RZ 3120 (#93166), IBM Research, April 1999. A revised version 4, dated November 15, 1999, is available from http://www.shoup.net/papers/.

[Yun02] Moti Yung, editor. *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Computer Science. Springer-Verlag, 18–22 August 2002.

Appendices

A Additional Definitions

A.1 Public Key Encryption

This subsection reviews the definition of security against adaptive chosen-ciphertext (CCA) attack[14]. The definitions in this section are essentially those of [BDPR98].

A public key cryptosystem consists of a triple of algorithms $\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ where:

$\mathcal{G}(1^k) \rightarrow \langle sk, pk \rangle$: the probabilistic key generation algorithm
- **In:** Security parameter k (in unary) [and other system parameters as needed]
- **Out:** Public and private key pair $\langle sk, pk \rangle$

$\mathcal{E}_{pk}(m) \rightarrow c$: the probabilistic encryption algorithm
- **In:** Public key pk, plaintext message m
- **Out:** Ciphertext c

$\mathcal{D}_{sk}(m) \rightarrow m'$: the deterministic decryption algorithm
- **In:** Secret key sk, ciphertext c
- **Out:** Plaintext message m' (where for all m, $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$), or a special symbol $\perp$ indicating the ciphertext c is invalid

Let $\mathcal{S} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public key cryptosystem and let $\mathcal{A}_{GM} = (\mathcal{A}_{probe}, \mathcal{A}_{dist})$ be an adversary consisting of a pair of probabilistic algorithms ($\mathcal{A}_{probe}$ is the probing component, $\mathcal{A}_{dist}$ is the distinguisher). Restrict $\hat{b}$ to be $\hat{b} \in \{0, 1\}$ and $|x_0| = |x_1|$. For a bit $b \in \{0, 1\}$ and security parameter k, define the following experiment:

Experiment $\mathbf{Exp}^{ind\text{–}cca\text{–}b}_{\mathcal{S},\mathcal{A}_{GM}}(k)$

```
$\langle sk, pk \rangle \xleftarrow{R} \mathcal{G}(1^k)$
$\langle x_0, x_1, s \rangle \leftarrow \mathcal{A}^{\mathcal{D}_{sk}(\cdot)}_{probe}(pk, 1^k)$ % generate the plaintext messages and state information $s$ to be passed to $\mathcal{A}_{dist}$
$y \leftarrow \mathcal{E}_{pk}(x_b)$
$\hat{b} \leftarrow \mathcal{A}^{\mathcal{D}_{sk}(\cdot)}_{dist}(x_0, x_1, s, y, 1^k)$
return $\hat{b}$
```

[14] Both here and in the channel security definition we don't distinguish between CCA1 and CCA2 security. We will only consider the stronger CCA2 definition.

We also require that $\mathcal{A}_{dist}$ does not ask its oracle to decrypt the ciphertext challenge y. Define the *advantage* of $\mathcal{A}_{GM}$ against $\mathcal{S}$ to be:

$$\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}_{GM}}(k) = Pr[\mathbf{Exp}^{ind\text{–}cca\text{–}1}_{\mathcal{S},\mathcal{A}_{GM}}(k) = 1] - Pr[\mathbf{Exp}^{ind\text{–}cca\text{–}0}_{\mathcal{S},\mathcal{A}_{GM}}(k) = 1].$$

Scheme $\mathcal{S}$ is *cca-secure* if $\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}_{GM}}(\cdot)$ is negligible[15] for all PPT $\mathcal{A}_{GM}$.

Auxiliary Definitions: Crucial Refresh; To- and Fro- Attacks. In each execution of the above experiment, for a given adversary challenge request $(u, \bar{u}, \tau, m_0, m_1)$, call u as the *challenge sender*, $\bar{u}$ as the *challenge receiver* and τ as the *challenge period*. We refer to $R^{[\bar{w}]}_{\tau-1}$ (where $\bar{w} = U[\tau - 1]$), the refresh message initiating the challenge period, as the *crucial refresh*.

We say that $\mathcal{A}$ executes a *to-attack* whenever $\mathcal{A}$ specifies that the crucial refresh be sent from the challenge sender *to* the challenge receiver, i.e. the crucial refresh direction is the same as the challenge direction ($\bar{w} = \bar{u}$). Corresponding to this, let the *to-attack advantage* of $\mathcal{A}$, denoted $\epsilon_{to}(k)$, be defined as:

$$\epsilon_{to}(k) \overset{def}{=} \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}1}_{\mathcal{KEC},\mathcal{A}}(k) = 1 \wedge \bar{w} = \bar{u}] - \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}0}_{\mathcal{KEC},\mathcal{A}}(k) = 1 \wedge \bar{w} = \bar{u}].$$

Inversely, we say that $\mathcal{A}$ executes a *fro-attack* whenever $\mathcal{A}$ specifies that the crucial refresh be sent *from* the challenge receiver to the challenge sender, i.e. the crucial refresh direction is opposite to the challenge direction ($\bar{w} = u$). Corresponding to this, let the *fro-attack advantage* of $\mathcal{A}$, denoted $\epsilon_{fro}(k)$, be defined as:

$$\epsilon_{fro}(k) \overset{def}{=} \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}1}_{\mathcal{KEC},\mathcal{A}}(k) = 1 \wedge \bar{w} = u] - \Pr[\mathbf{Exp}^{channel\text{–}ind\text{–}0}_{\mathcal{KEC},\mathcal{A}}(k) = 1 \wedge \bar{w} = u].$$

In this way we have partitioned the set of all possible executions of $\mathbf{Exp}^{channel\text{–}ind\text{–}1}_{\mathcal{KEC},\mathcal{A}}(k)$ into two disjoint sets based upon the direction in which the crucial refresh is sent, so that

$$\epsilon(k) = \epsilon_{to}(k) + \epsilon_{fro}(k)\,.$$

This classification of the advantage of $\mathcal{A}$ will be useful in proving the intrusion-resilience of the channel construction detailed in Section 3.

A.2 Channel Spliceability

In formulating the following channel properties we assume that the $\mathcal{KEC}$ keys for party u consist of four components: the incoming and outgoing data and refresh sub-channel keys. These are denoted $SK^{[u]}_{\tau}.pk^{[\bar{u}|d\rangle}_{\tau}$, $SK^{[u]}_{\tau}.sk^{[u|d\rangle}_{\tau}$, $SK^{[u]}_{\tau}.pk^{[\bar{u}|\rho\rangle}_{\tau}$, and $SK^{[u]}_{\tau}.sk^{[u|\rho\rangle}_{\tau}$, respectively. Party $\bar{u}$ has similar keys with u and $\bar{u}$ switched in the above.

In the splice experiment $\mathbf{Exp}^{ind\text{–}chan\text{–}splice\text{–}b}_{\mathcal{KEC},\mathcal{A}}(k)$ below, the distinguishing adversary $\mathcal{A}$ interacts with a channel instance and selects a "test" period and direction. If the bit $b = 1$, a fresh random key pair is "spliced" into the channel for the test period, replacing the outgoing data sub-channel keys of that period's refresh sender (details under $ChangeFMKey(\cdot)$ below). Encryption queries corresponding to the refresh sender in the test period are still answered using the original key,

[15] A function $\nu(k)$ is *negligible* if for any polynomial $p(k)$ and all sufficiently large k, $\nu(k) < \frac{1}{p(k)}$.

but a key exposure of the refresh sender reveals the alternate key. Resulting consistency issues are handled by the decryption oracle. If $b = 0$, no splicing is done. Recall that $U[\tau]$ contains the identity of the refresh receiver for period τ. A detailed description of the changes to the channel key exposure and decryption oracles, $Okey_t$ and $Odec_t$, from Section 2.2 follows.

```
Oracle $Okey_{b,t,\tau}(u, \alpha)$
    if $b = 1$ and $\alpha = \tau$ and $U[\tau - 1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed
        then return $C_1.SK_\tau^{[u]}$        % *expose alternate $C_1$ channel sender key*
    else return $Okey_t(u, \alpha)$        % *expose primary channel key*

Oracle $Odec_{b,t,\tau}(c, \bar{u}, \alpha)$
    if $b = 1$ and $\alpha = \tau$ and $U[\tau - 1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed
        and $Oenc_{b,t,\tau}$ does not contain $(\alpha, u, \cdot, c)$ % *gave alt. key and c not from Oenc*
        then return $Odec_{C_1.SK_\tau^{[\bar{u}]}}(c)$    % *enc. oracle not used; decrypt using alt. chan key*
    else return $Odec_t(c, \bar{u}, \alpha)$  % *use usual channel decryption oracle*

Algorithm $ChangeFMKey(SK_\tau^{[u]})$
    $\langle SK^{[0]}, SK^{[1]} \rangle \leftarrow \mathcal{KEC}.InitKeys(1^k[, \ldots])$  % *generate new channel key pair*
    $C_1.SK_\tau^{[u]} \leftarrow SK_\tau^{[u]}$  % *copy primary channel u key to alternate channel u key*
    $C_1.SK_\tau^{[u]}.pk^{[\bar{u}|d\rangle} \leftarrow SK^{[u]}.pk^{[\bar{u}|d\rangle}$    % *replace u's outgoing data message key*
    $C_1.SK_\tau^{[\bar{u}]}.sk^{[\bar{u}|d\rangle} \leftarrow SK^{[\bar{u}]}.sk^{[\bar{u}|d\rangle}$    % *replace ū's incoming data message key*
    return $\langle C_1.SK_\tau^{[u]}, C_1.SK_\tau^{[\bar{u}]} \rangle$
```

The full splice experiment is given below, in which $O_{b,t}$ denotes the entire set of oracles to which $\mathcal{A}$ has access during period t.

```
Experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}splice\text{-}b}(k)$
INIT:
    $t \leftarrow 0$; $\tau \leftarrow \perp$
    $\langle SK_t^{[0]}, SK_t^{[1]} \rangle \leftarrow InitKeys(1^k)$  % *initialize primary chan. keys*
PROBE:
    $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$
    while $q$ in $(\mathrm{ref}(w, \bar{w}), \mathrm{test}(w, \bar{w}))$ do  % *send refresh from party w to party w̄*
        $\langle SK_{t+1}^{[w]}, R_t^{[\bar{w}]} \rangle \leftarrow GenR(SK_t^{[w]})$ % *party w generates new key and refresh*
        $SK_{t+1}^{[\bar{w}]} \leftarrow ProcR_{SK_t^{[\bar{w}]}}(R_t^{[\bar{w}]})$  % *party w̄ receives and processes refresh*
        if $b = 1$ and $q = \mathrm{test}(w, \bar{w})$ and $\tau = \perp$ then  % *generate alternate channel key pair only once*
            $\langle C_1.SK_{t+1}^{[w]}, C_1.SK_{t+1}^{[\bar{w}]} \rangle \leftarrow ChangeFMKeys(SK_{t+1}^{[w]})$
            $\tau \leftarrow t + 1$  % *record challenge period*
        $U[t] \leftarrow \bar{w}$  % *party w̄ recorded as period t refresh receiver*
        $t \leftarrow t + 1$  % *time period incremented*
        $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$
DIST:
    $s \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$  % $\mathcal{A}_{probe}$ outputs state $s$
    $\hat{b} \leftarrow \mathcal{A}_{dist}^{O_{b,t}}(1^k, s)$  % $\mathcal{A}_{dist}$ outputs distinguishing guess $\hat{b}$ for $b$, i.e. plain vs. single splice channel
    return $\hat{b}$
```

Definition 9 (Spliceability).
Let $\mathcal{KEC} = (InitKeys, EncM, DecM, GenR, ProcR; \underline{SendMesg}, \underline{RefKeys})$ be a two-party key evolving channel scheme with security parameter k, and let $\mathcal{A}$ be an adversary. Define the (CCA) splice-distinguishing advantage *of $\mathcal{A}$ against the channel $\mathcal{KEC}$ as:*

$$\mathbf{Adv}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}splice}(k) \overset{def}{=} \Pr[\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}splice\text{–}0}(k) = 1] - \Pr[\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}splice\text{–}1}(k) = 1]\,.$$

$\mathcal{KEC}$ is (CCA) splice-indistinguishable[16] *or just* spliceable *if $\mathbf{Adv}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}splice}(\cdot)$ is negligible for all refresh-receiver-biased PPT $\mathcal{A}$.*

In the above experiment the refresh sender's key is changed for a *single* period of $\mathcal{A}$'s choosing. One can similarly define a *multi*-splice version $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}Msplice\text{–}b}(k)$, in which keys of the refresh sender are changed for *all* periods in which the corresponding refresh receiver is not exposed. In this case the key exposure and decryption oracles become:

```
Oracle $Okey_{b,t}(u, \tau)$
    if $b = 1$ and $U[\tau - 1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed
     % $u$ is a forbidden message sender for period $\tau$
    then return $C_1.SK_\tau^{[u]}$                    % expose alternate $C_1$ channel key
    else return $Okey_t(u, \tau)$                     % expose primary channel key


Oracle $Odec_{b,t}(c, \bar{u}, \tau)$
    if $b = 1$ and $U[\tau - 1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed
        and $Oenc_t$ does not contain $(\tau, u, \cdot, c)$
     % $u$ is a forbidden message sender for period $\tau$ and didn't use oracle to create $c$
    then return $Odec_{C_1.SK_\tau^{[\bar{u}]}}(c, \tau)$  % use alternate $C_1$ channel decryption oracle
    else return $Odec_t(\tau, \bar{u}, c)$  % use primary decryption oracle
```

The full multi-splice experiment is given below. As in the splice experiment above, $O_{b,t}$ denotes the entire set of oracles to which $\mathcal{A}$ has access during period t.

```
Experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}Msplice\text{–}b}(k)$
INIT:
    $t \leftarrow 0$
    $\langle SK_t^{[0]}, SK_t^{[1]} \rangle \leftarrow InitKeys(1^k)$                % initialize primary chan. keys
PROBE:
    $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$
```

[16] A broader definition of forbidden messages is possible. Namely, any message whose receiver is not exposed could be considered forbidden. However, proving that a channel is secure for this broader class of forbidden messages seems difficult.

```
    while $q$ in $(\mathrm{ref}(w,\bar{w}))$ do                              % send refresh from party $w$ to party $\bar{w}$
        $\langle SK_{t+1}^{[w]}, R_t^{[\bar{w}]}\rangle \leftarrow GenR(SK_t^{[w]})$      % party $w$ generates new key and refresh
        $SK_{t+1}^{[\bar{w}]} \leftarrow ProcR_{SK_t^{[\bar{w}]}}(R_t^{[\bar{w}]})$          % party $\bar{w}$ receives refresh and updates its key
        $\langle C_1.SK_t^{[w]}, C_1.SK_t^{[\bar{w}]}\rangle \leftarrow ChangeFMKeys(SK_t^{[w]})$
        $U[t] \leftarrow \bar{w}$                                         % party $\bar{w}$ is recorded as refresh receiver for period $t$
        $t \leftarrow t+1$                                             % time period incremented
        $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$
  DIST:
    $s \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$   % $\mathcal{A}_{probe}$ outputs state $s$
    $\hat{b} \leftarrow \mathcal{A}_{dist}^{O_{b,t}}(1^k, s)$   % $\mathcal{A}_{dist}$ outputs distinguishing guess $\hat{b}$ for $b$, i.e. plain or spliced channel
    return $\hat{b}$
```

Definition 10 (Spliceability (multiple period)).
Let $\mathcal{KEC} = (InitKeys, EncM, DecM, GenR, ProcR)$ be a two-party key evolving channel scheme with security parameter k, and let $\mathcal{A}$ be an adversary. We define the (CCA) multi-splice-distinguishing advantage *of $\mathcal{A}$ against the channel $\mathcal{KEC}$ as:*

$$\mathbf{Adv}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}Msplice}(k) \overset{def}{=} \Pr[\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}Msplice\text{-}0}(k)=1] - \Pr[\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}Msplice\text{-}1}(k)=1]\,.$$

$\mathcal{KEC}$ is (CCA) multi-splice-indistinguishable *or just* multi-spliceable *if $\mathbf{Adv}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}Msplice}(\cdot)$ is negligible for all refresh-receiver-biased PPT $\mathcal{A}$.*

The following lemma is used in the proof of Theorem 3, our channel composition result.

Lemma 4 (Spliceable ⇒ Multi-Spliceable). *If $\mathcal{KEC}$ is spliceable, then it is also multi-spliceable.*

Proof. Assume there exists a multi-splice adversary $\mathcal{A}$ with noticeable advantage ε against $\mathcal{KEC}$. We show via a hybrid argument that this implies the existence of a splice adversary $\mathcal{A}_1$ with noticeable advantage against $\mathcal{KEC}$. $\mathcal{A}_1$ initially guesses a hybrid period $1 \leq \tau \leq T$, and then simulates a hybrid version of experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{-}chan\text{-}Msplice\text{-}b}(k)$. In this hybrid experiment, all forbidden message sender keys for periods $< \tau$ are spliced, all keys for periods $> \tau$ are not spliced (left unchanged), and the key of a period τ forbidden message sender will be the key for the splice challenge period in the experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}_1}^{ind\text{-}chan\text{-}splice\text{-}b}(k)$.

$\mathcal{A}_1$ handles the oracle queries of $\mathcal{A}$ as follows. The only special cases are $\mathcal{A}$'s queries for encryptions, key exposures, or decryptions for time periods $< \tau$. For all other queries, $\mathcal{A}_1$ simply forwards $\mathcal{A}$'s query to its own corresponding oracle, and hands the answer back to $\mathcal{A}$ unchanged.

To handle an encryption query (m, u, α) for some period $\alpha < \tau$, $\mathcal{A}_1$ still forwards the query to its own oracle and returns the resulting ciphertext c to $\mathcal{A}$ unchanged. The only thing different is that $\mathcal{A}_1$ adds this query-response pair $((m, u, \alpha), c)$ to a running log of encryption queries made by $\mathcal{A}$ for later use when answering decryption queries.

To handle a key exposure query (u, α) for some period $\alpha < \tau$, $\mathcal{A}_1$ checks to see whether party u was the sender of the refresh initiating period α and whether the corresponding refresh receiver is still unexposed. If either one of these conditions fails to hold, $\mathcal{A}_1$ forwards the query to its own oracle and returns the answer unchanged to $\mathcal{A}$. Otherwise, if both conditions hold, party u is a forbidden message sender for period α. $\mathcal{A}_1$ still forwards the exposure query to its own oracle. However, after receiving the answer $SK_\alpha^{[u]}$, instead of passing this key unchanged to $\mathcal{A}$, as in Experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}Msplice\text{–}b}(k)$ $\mathcal{A}_1$ uses this key to generate alternate keys $C_1.SK_\alpha^{[u]}$ and $C_1.SK_\alpha^{[\bar{u}]}$. $\mathcal{A}_1$ gives $C_1.SK_\alpha^{[u]}$ to $\mathcal{A}$ and stores $C_1.SK_\alpha^{[\bar{u}]}$ for later use when answering decryption oracle queries.

To handle a decryption query $(c, \bar{u}, \alpha)$ for some period $\alpha < \tau$, $\mathcal{A}_1$ first checks to see whether u is a forbidden message sender for period α. If u is not a forbidden message sender for period α, or if this is not yet determined (i.e. neither u's nor $\bar{u}$'s keys for period α have been exposed yet), then $\mathcal{A}_1$ simply forwards the query to its own oracle and returns the answer unchanged to $\mathcal{A}$. Otherwise, if party u is a forbidden message sender for period α, then $\mathcal{A}_1$ first checks its running log to see if the ciphertext c was created via an encryption oracle query of $\mathcal{A}$. If the log contains some entry $((m, u, \alpha), c)$, then $\mathcal{A}_1$ just returns the message m from this log entry directly to $\mathcal{A}$ (the same answer as would be obtained by querying the decryption oracle). If on the other hand the log does not contain such an entry, $\mathcal{A}_1$ decrypts c using the alternate party $\bar{u}$ key $C_1.SK_\alpha^{[\bar{u}]}$ for period α[17].

When $\mathcal{A}$ outputs its distinguishing guess $\hat{b} \in \{0,1\}$, $\mathcal{A}_1$ halts and outputs $\hat{b}$ as its own guess.

Finally, note that if $\mathcal{A}$ has noticeable advantage ε in $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}Msplice\text{–}b}(k)$, then by the above hybrid simulation, $\mathcal{A}_1$ has noticeable advantage of roughly ε/T in $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}_1}^{ind\text{–}chan\text{–}splice\text{–}b}(k)$.

A.3 Channel Key-Message-Splice Indistinguishability

This property captures the notion that for a single period during which the refresh receiver is not exposed, no PPT adversary should be able to distinguish between the following two worlds. World 1 is identical to world 1 from the splice experiment above: the corresponding refresh sender's key has been changed in the test period. In world 0, the refresh sender's key has not been changed in the test period, but whenever the adversary queries for an encryption by the refresh sender for the test period, he is given an encryption of randomness (under the correct key). For this experiment, the key exposure, encryption and decryption oracles from Section 2.2 become:

```
Oracle $Okey_{b,t,\tau}(u, \alpha)$
    if $b = 1$ and $\alpha = \tau$ and $U[\tau - 1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed
    then return $C_1.SK_\tau^{[u]}$          % expose alternate $C_1$ channel key
    else return $Okey_t(u, \alpha)$          % expose primary channel key


Oracle $Oenc_{b,t,\tau}(m, u, \alpha)$
    if $b = 0$ and $\alpha = \tau$ and $U[\tau - 1] = \bar{u}$
        $r \xleftarrow{R} \{0,1\}^{|m|}$
        return $Oenc_t(r, u, \alpha)$   % encrypt randomness instead of message
    else return $Oenc_t(m, u, \alpha)$   % encrypt message
```

[17] $\mathcal{A}_1$ decrypts c with the alternate key in this case since it knows that $\mathcal{A}$ did not generate c via an encryption oracle query. This is done to ensure consistency since $\mathcal{A}$ may have produced c using the alternate party u key $C_1.SK_\alpha^{[u]}$ provided by $\mathcal{A}_1$. Consequently, $\mathcal{A}_1$'s decryption oracle in general will not produce the correct answer.

```
Oracle $Odec_{b,t,\tau}(c,\bar{u},\alpha)$
    if $\alpha = \tau$ and $U[\tau-1] = \bar{u}$ and $SK_\tau^{[\bar{u}]}$ is not $Q$-exposed then
        CASE $b$
            0:
                if $Oenc_{b,t,\tau}$ contains some $(\alpha, u, m, c)$ % enc. oracle used
                    then return $m$ % return logged message instead
                else return $Odec_t(c, \bar{u}, \alpha)$ % use usual chan. decrypt. oracle
            1:
                if $Oenc_{b,t,\tau}$ contains some $(\alpha, u, \cdot, c)$ % enc. oracle used
                    then return $Odec_t(c, \bar{u}, \alpha)$ % use usual chan. decrypt. oracle
                else return $Odec_{C_1.SK_\tau^{[\bar{u}]}}(c)$ % decrypt using alt. key
    else return $Odec_t(c, \bar{u}, \alpha)$ % use usual channel decryption oracle
```

```
Experiment $\mathbf{Exp}^{ind\text{-}chan\text{-}KeyMsg\text{-}splice\text{-}b}_{\mathcal{KEC},\mathcal{A}}(k)$
INIT:
    $t \leftarrow 0$; $\tau \leftarrow \perp$; $\langle SK_t^{[0]}, SK_t^{[1]} \rangle \leftarrow InitKeys(1^k)$ % initialize primary channel keys
PROBE:
    $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$
    while $q$ in $(\mathrm{ref}(w,\bar{w}), \mathrm{test}(w,\bar{w}))$ do % send refresh from party $w$ to party $\bar{w}$
        $\langle SK_{t+1}^{[w]}, R_t^{[\bar{w}]} \rangle \leftarrow GenR(SK_t^{[w]})$ % party $w$ generates new key and refresh
        $SK_{t+1}^{[\bar{w}]} \leftarrow ProcR_{SK_t^{[\bar{w}]}}(R_t^{[\bar{w}]})$ % party $\bar{w}$ receives and processes refresh
        if $q = \mathrm{test}(w,\bar{w})$ and $\tau = \perp$ then
            $\tau \leftarrow t+1$ % record challenge period
            if $b = 1$ then % generate alternate channel key pair only once
                $\langle C_1.SK_{t+1}^{[w]}, C_1.SK_{t+1}^{[\bar{w}]} \rangle \leftarrow ChangeFMKeys(SK_{t+1}^{[w]})$
        $U[t] \leftarrow \bar{w}$ % party $\bar{w}$ recorded as period $t$ refresh receiver
        $t \leftarrow t+1$; $q \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$ % increment time period; get next query

DIST:
    $s \leftarrow \mathcal{A}_{probe}^{O_{b,t}}(1^k)$ % $\mathcal{A}_{probe}$ outputs state $s$
    $\hat{b} \leftarrow \mathcal{A}_{dist}^{O_{b,t}}(1^k, s)$ % $\mathcal{A}_{dist}$ outputs distinguishing guess $\hat{b}$ for $b$, i.e. plain vs. single splice channel
    return $\hat{b}$
```

NOTE: $\mathcal{A}$ is not allowed to key expose receiver of refresh in period τ. In the $b = 0$ case $\mathcal{A}$ could use key to decrypt altered ciphertexts.

Definition 11 (Key-Message-Splice-Indistinguishability).
Let $\mathcal{KEC} = (InitKeys, EncM, DecM, GenR, ProcR)$ be a two-party key-evolving channel scheme with security parameter k, and let $\mathcal{A}$ be an adversary. We define the (CCA) Key-Message-Splice distinguishing advantage *of $\mathcal{A}$ against the channel $\mathcal{KEC}$ as:*

$$\mathbf{Adv}^{ind\text{-}chan\text{-}KeyMsg\text{-}splice}_{\mathcal{KEC},\mathcal{A}}(k) \stackrel{def}{=} \Pr[\mathbf{Exp}^{ind\text{-}chan\text{-}KeyMsg\text{-}splice\text{-}0}_{\mathcal{KEC},\mathcal{A}}(k)=1] - \Pr[\mathbf{Exp}^{ind\text{-}chan\text{-}KeyMsg\text{-}splice\text{-}1}_{\mathcal{KEC},\mathcal{A}}(k)=1] .$$

We say $\mathcal{KEC}$ is (CCA) Key-Message-Splice-Indistinguishable *if* $\mathbf{Adv}^{ind\text{–}chan\text{–}KeyMsg\text{–}splice}_{\mathcal{KEC},\mathcal{A}}(\cdot)$ *is negligible for all refresh-receiver-biased PPT $\mathcal{A}$.*

Lemma 5. *If $\mathcal{KEC}$ is spliceable and intrusion resilient, then it is Key-Message-Splice-Indistinguishable.*

Proof. For the sake of contradiction, suppose that there exists an $\mathcal{A}$ with noticeable distinguishing advantage in $\mathbf{Exp}^{ind\text{–}chan\text{–}KeyMsg\text{–}splice\text{–}b}_{\mathcal{KEC},\mathcal{A}}$. Observe that the only difference in $\mathcal{A}$'s view between the case of $b = 0$ and $b = 1$ occurs in the test period τ. Let u denote the sender of the refresh, $R^{[\bar{u}]}_{\tau-1}$, that initiates period τ, and for $1 \leq i \leq N$ let m_i be the message in the ith u encryption query for period τ. Then $\mathcal{A}$ must be able to distinguish between:

$$\left(R^{[u]}_{\tau-1}, SK^{[u]}_{\tau}, \vec{m}, \overrightarrow{EncM_{SK^{[u]}_{\tau}}(r)}, \vec{c}, \overrightarrow{Odec_{0,t,\tau}(c)}\right), \text{and} \tag{1}$$

$$\left(R^{[u]}_{\tau-1}, C_1.SK^{[u]}_{\tau}, \vec{m}, \overrightarrow{EncM_{SK^{[u]}_{\tau}}(m)}, \vec{c}, \overrightarrow{Odec_{1,t,\tau}(c)}\right) \tag{2}$$

where $r_i \overset{R}{\leftarrow} \{0,1\}^{|m_i|}$. But if this is the case, then by the triangle inequality $\mathcal{A}$ must be able to distinguish either between $(\dagger) = \left(R^{[u]}_{\tau-1}, SK^{[u]}_{\tau}, \vec{m}, \overrightarrow{EncM_{SK^{[u]}_{\tau}}(m)}, \vec{c}, \overrightarrow{DecM_{SK^{[\bar{u}]}_{\tau}}(c)}\right)$ and (2), or between $(\dagger)$ and (2). Thus, such an $\mathcal{A}$ can by used in the former case to build a splice adversary against $\mathcal{KEC}$ and in the latter case to build via a hybrid argument an intrusion-resilience adversary against $\mathcal{KEC}$. In either case this results in a contradiction.

B Intrusion-Resilient Model

B.1 Model Review

Actor, Base, Refreshes. Let us consider the intrusion-resilient model. There, some cryptographic function (e.g., signature generation) is performed by an entity which we here call *actor* (in [IR02,Itk02], the actor is referred to as Signer). However, the secret key information maintained by the actor expires periodically, at which time he needs to receive an *update* message from another entity (*base*). The life of the system is divided into *time periods* according to these updates: each update signals the beginning of the new time period. In addition, the base may send *refresh* messages to the actor — as frequently and as many as desired.

Exposures. Clearly, if an adversary exposes the actor during some time period, she can do everything during that period that the actor could (e.g., generate signatures for the period of the exposure). However, at the end of the time period, the exposed keys expire. The base will then issue an update message. If the adversary does not receive this message then she will no longer enjoy any benefit of the exposure. The exposures of the base by themselves are of no consequence.

Simultaneous Exposure. Of course, if the adversary manages to expose all the information of the system —i.e., by exposing both the base and the actor simultaneously— then she can perform all the functions of the system from that point onwards (one potential limitation on the adversary even in this case is discussed in [Itk03]).

Even in this case of simultaneous exposure, intrusion-resilient systems preserve forward-security: the system remains secure for the periods prior to the exposure (e.g., signatures cannot be backdated and old messages cannot be deciphered).

Direct and Indirect Exposures. Similar to the definitions of exposure in Sec. 2.2, there are likewise two types of exposures in the traditional intrusion-resilient model: direct and indirect. The actor's secret key can be indirectly exposed in a way similar to exposure via evolution as defined in Sec. 2.2. The past definitions of intrusion-resilient models, however, also included a separate indirect exposure of the base, also corresponding to exposure via evolution. While the actor (receiver) evolution exposure is unavoidable, the base (sender) evolution exposure can be eliminated, as discussed below. This would effectively restrict the exposures of the base to direct only.

B.2 Motivations for Intrusion-Resilient Channels

Let us consider what exactly is meant by "simultaneous" exposures above. In the previous definitions of intrusion-resilient models, two exposures were considered simultaneous if there was no refresh between them. As a result, in the past models, the security of the system can be recovered as long as there is at least one refresh message not seen by the adversary between any two break-ins (direct exposures). Thus, the users of the system can make the simultaneous exposure more difficult by increasing the frequency of refreshes as deemed necessary.

Refresh Vulnerability. Still, this need to rely on the adversary missing some messages gave rise to a valid criticism of the model: it seems reasonable to assume that mounting an exposure (of either base or the actor) is a more demanding task than intercepting their messages. The past responses to this criticism were limited to appealing to "out-of-band" methods, e.g., to bring the actor into a physical contact with the base to secure transmission of refresh message. While this in some situations may indeed be an acceptable or even desirable feature, using and intrusion-resilient channel $\mathcal{IRC}$ offers a better option.

Let the base and actor use $\mathcal{IRC}$ to protect their communication. There are a number of benefits of doing so which are discussed at length next.

Eliminating Indirect Exposures Of The Base. First, in the past intrusion-resilient models it was possible to expose a base not only directly but also by using refresh or update messages it sent. With $\mathcal{IRC}$-secured communication, only direct exposures are possible for the base (except in some cases following a simultaneous exposure — where the issue is already irrelevant).

Eliminating Leaking Unexposed Information. Secondly, the original definitions of exposures allowed for the cases when the adversary knows a secret key, while that key is considered not exposed. This strange circumstance did not harm the specific models and their security properties, but it is counter-intuitive and creates a certain potential lack of robustness. Using $\mathcal{IRC}$ we eliminate this strange aberration, creating a tighter link between the concept of exposed keys and the knowledge of the adversary.

Tolerating More Powerful Adversary. Thirdly, some intrusion-resilient constructions needed to limit slightly the power of adversary in order to guarantee security. For example, the adversary could access only the very recent refresh messages (in a simulation). Some of such restrictions can be relaxed if the $\mathcal{IRC}$ is used.

Uni-Directional vs. Two-Directional $\mathcal{IRC}$. Finally, last, but not least: all three benefits above apply even if the $\mathcal{IRC}$ is restricted to send all messages —data and channel maintenance— in only one direction: from the base to the actor. The unidirectional nature of the intrusion-resilient model was motivated by the following consideration. The base and the signer together maintain some long term secret (even if it is evolving). For any implementation of a cryptographic module, the ability to observe the module react to various inputs (or even provide these inputs) represents a significant power for the adversary. Clearly, the actor cannot avoid this threat. However, it is possible to restrict the base to accept no inputs, thus raising its, and therefore system's, level of security.

In contrast, $\mathcal{IRC}$ maintains no long term secrets. Therefore, the above motivation for unidirectionality does not apply to it. This may justify the ability of the $\mathcal{IRC}$ to support two-directional channel refresh messages (even if the data traffic is still only from the base to the actor). This strengthens the model even further, since the channel recovers even if its channel refresh message is intercepted. In other words, the window of vulnerability for indirect exposure of the actor is limited to the interval between the last $\mathcal{IRC}$ refresh sent by the actor's end of the channel and the subsequent refresh sent by the base. Since this interval can be made arbitrarily small, this practically eliminates even the indirect exposures of the actor: the only way then to expose either party is a direct exposure.

C Channel Security: Proof of Theorem 1 ($g\mathcal{IRC}$ is Intrusion-Resilient)

The following proof of Theorem 1 relies on Lemma 6 and Corollary 1, which in turn comes from Lemmas 7 and 8. These Lemmas and Corollary are precisely stated in section C.1. Lemma 6 deals with the case of a fro-attacking channel adversary, while the case of a to-attacking channel adversary is dealt with in Lemmas 7 and 8. The proofs of these Lemmas and Corollary appear in Appendices C.2–C.6.

Proof (of Theorem 1). For a given PPT channel adversary $\mathcal{A}$, let $\epsilon(k) = \mathbf{Adv}^{channel\text{–}ind}_{g\mathcal{IRC},\mathcal{A}}(k) = \epsilon_{fro}(k) + \epsilon_{to}(k)$ as defined at the end of Section 2.2. Then Lemma 6 implies that there exists a PPT adversary $\mathcal{A}_{GM}$ with advantage $\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}_{GM}}(k) = \frac{\epsilon_{fro}(k)}{q_{refr}}$ against $\mathcal{S}$. Also, Corollary 1 implies that there exists a pair of PPT adversaries $(\mathcal{A}^{rand}_{GM}, \mathcal{A}^{hyb}_{GM})$ such that

$$\max\{\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k), \mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k)\} \geq \frac{\epsilon_{to}(k)}{2{q_{refr}}^2(3q_{refr}-2)} .$$

Since $\epsilon(k) = \epsilon_{fro}(k) + \epsilon_{to}(k)$, it follows that either $\epsilon_{fro}(k) \geq \frac{\epsilon(k)}{2}$ or $\epsilon_{to}(k) \geq \frac{\epsilon(k)}{2}$. Thus, either $\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}_{GM}}(k) \geq \frac{\epsilon(k)}{2q_{refr}}$ or $\max\{\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k), \mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k)\} \geq \frac{\epsilon(k)}{4{q_{refr}}^2(3q_{refr}-2)}$.

Using $q_{refr} \geq 1 \Rightarrow \frac{\epsilon(k)}{2q_{refr}} \geq \frac{\epsilon(k)}{4{q_{refr}}^2(3q_{refr}-2)}$, the above yields that for each value of k at least one adversary of the triple $(\mathcal{A}_{GM}, \mathcal{A}^{rand}_{GM}, \mathcal{A}^{hyb}_{GM})$ will have advantage $\geq \frac{\epsilon(k)}{4{q_{refr}}^2(3q_{refr}-2)}$ against $\mathcal{S}$.

Finally, note that the running times of the three $\mathcal{S}$-adversaries above are essentially the same as $\mathcal{A}$. □

C.1 Lemma and Corollary Statements

In the Lemma statements below let $\mathcal{A}$, q_{refr}, $\epsilon(k)$, $\epsilon_{to}(k), \epsilon_{fro}(k)$ all be defined as above.

The Fro-Attack Case In the case of a fro-attack the only part of the challenge receiver's key to appear in the crucial fro-refresh message is his new public key. Consequently, it is the easier case to simulate: the $\mathcal{S}$-adversary $\mathcal{A}_{GM}$ when lucky enough to guess the challenge period correctly can easily slip the given public key into the crucial refresh and thus trick $\mathcal{A}$ into answering the challenge. The following lemma gives the advantage of $\mathcal{S}$-adversary $\mathcal{A}_{GM}$ in terms of channel adversary $\mathcal{A}$'s fro-attack advantage.

Lemma 6 **($\exists\mathcal{A} \Rightarrow \exists\mathcal{A}_{GM}$).**
Let $\mathcal{A}$ be any $g\mathcal{IRC}$-adversary causing at most q_{refr} `PROBE` loop iterations in $\mathbf{Exp}^{channel\text{-}ind\text{-}b}_{g\mathcal{IRC},\mathcal{A}}(k)$ and having fro-*attack advantage $\epsilon_{fro}(k)$.*
Then there exists an $\mathcal{S}$-adversary $\mathcal{A}_{GM}$ such that $\mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}_{GM}}(k) = \frac{\epsilon_{fro}(k)}{q_{refr}}$.
Furthermore, $\mathcal{A}_{GM}$ runs $\mathcal{A}$ as an oracle and performs at most q_{refr} $g\mathcal{IRC}$.GenKeys operations, and one $g\mathcal{IRC}$.EncM or $g\mathcal{IRC}$.DecM operation or Odec query per each Oenc or Odec query made by $\mathcal{A}$; all other work done by $\mathcal{A}_{GM}$ does not exceed $O(1)$ per each of the above operations.

The To-Attack Case In contrast to the case of a fro-attack, in a to-attack the challenge receiver's secret key for the challenge period appears in the crucial to-refresh message. Consequently, this is the harder case to simulate: even when lucky enough to guess the challenge period correctly, an $\mathcal{S}$-adversary $\mathcal{A}_{GM}$ cannot properly construct the crucial refresh message. To get around this difficulty, we first define a weaker $\mathcal{S}$-adversary $\mathcal{A}_{Chain}$ which receives a chained sequence of public key pairs and ciphertexts. For each element in the chain, the pair of public keys corresponds to the data and refresh public keys of a new channel period. Likewise the ciphertext of a given chain element corresponds to the to-refresh message initiating the new channel period; its plaintext consists of the new data and refresh secret keys, together with an arbitrary suffix padding chosen by $\mathcal{A}_{Chain}$. As in the channel setting, each successive ciphertext (to-refresh) in the chain is encrypted with the previous refresh public key. $\mathcal{A}_{Chain}$'s goal is to pick an instance in the chain against whose data secret key it will attempt to distinguish ciphertexts. We refer to such an adversary as a *chain* adversary and detail its security experiment in Appendix C.3.

Lemma 7, stated below, shows that there exists a chain adversary who *can* perfectly simulate for a given to-attacking channel adversary and thus capitalize on its to-attack advantage.

Lemma 7 **($\exists A \Rightarrow \exists\mathcal{A}_{Chain}$).**
Let $\mathcal{A}$ be any $g\mathcal{IRC}$-adversary causing at most q_{refr} `PROBE` loop iterations in $\mathbf{Exp}^{channel\text{-}ind\text{-}b}_{g\mathcal{IRC},\mathcal{A}}(k)$ and having to-*attack advantage $\epsilon_{to}(k)$. Then there exists a chain $\mathcal{S}$-adversary $\mathcal{A}_{Chain}$ such that $\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k) \geq \frac{\epsilon_{to}(k)}{2{q_{refr}}^2}$.*
Furthermore, $\mathcal{A}_{Chain}$ using $\mathcal{A}$ as an oracle obtains at most q_{refr} chain elements in $\mathbf{Exp}^{chain\text{-}ind\text{-}b}_{\mathcal{S},\mathcal{A}_{Chain}}(k)$, performs at most q_{refr} executions of $g\mathcal{IRC}$.GenKeys, and one $g\mathcal{IRC}$.EncM or $g\mathcal{IRC}$.DecM execution or Odec query per each Oenc or Odec query made by $\mathcal{A}$; all other work done by $\mathcal{A}_{Chain}$ does not exceed $O(1)$ per each of the above operations.

Finally, Lemma 8 below shows that an ordinary $\mathcal{S}$-adversary can be constructed from a given chain adversary and realize comparable advantage.

Lemma 8 **($\exists\mathcal{A}_{Chain} \Rightarrow \exists\mathcal{A}_{GM}$).**
Let $\mathcal{A}_{Chain}$ be any chain $\mathcal{S}$-adversary obtaining at most q_{next} chain elements in $\mathbf{Exp}^{chain\text{-}ind\text{-}b}_{\mathcal{S},\mathcal{A}_{Chain}}(k)$.

Then there exists a pair of $\mathcal{S}$-adversaries $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$ such that $\max\{\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{–}cca}(k), \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca}(k)\} \geq \left(\frac{1}{3q_{next}-2}\right) \cdot \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind}(k)$.
Furthermore, both $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$ use $\mathcal{A}_{Chain}$ as an oracle, perform at most $2q_{next}$ executions of $\mathcal{S}.\mathcal{G}$, q_{next} executions of $\mathcal{S}.\mathcal{E}$, and one $\mathcal{S}.\mathcal{D}$ execution or Odec query per each Odec query made by $\mathcal{A}_{Chain}$; all other work done by $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$ does not exceed $O(1)$ per each of the above operations.

The following Corollary composes the above pair of lemmas:

Corollary 1. *Let $\mathcal{A}$ be any $g\mathcal{IRC}$-adversary adversary with* to-*attack advantage $\epsilon_{to}(k)$ causing at most q_{refr}* `PROBE` *loop iterations in $\mathbf{Exp}_{g\mathcal{IRC},\mathcal{A}}^{channel\text{–}ind\text{–}b}(k)$. Then there exists a pair of adversaries $(\mathcal{A}_{GM}^{rand}, \mathcal{A}_{GM}^{hyb})$ such that* $\max\{\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{–}cca}(k), \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca}(k)\} \geq \frac{\epsilon_{to}(k)}{2q_{refr}{}^2(3q_{refr}-2)}$.
Furthermore, both $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$ use $\mathcal{A}$ as an oracle and perform at most q_{refr} executions of $g\mathcal{IRC}.GenKeys$ and one $g\mathcal{IRC}.EncM$ or $g\mathcal{IRC}.DecM$ execution or Odec query per each Oenc or Odec query made by $\mathcal{A}$; all other work done by $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$ does not exceed $O(1)$ per each of the above operations.

The proofs of the above Lemmas and Corollary appear in Appendices C.2–C.6.

C.2 The Fro-Attack Case: Proof of Lemma 6

Proof. Let $\hat{pk}$ denote the public key for the $\mathcal{S}$ instance being attacked by CCA-adversary $\mathcal{A}_{GM}$. Hence, $\mathcal{A}_{GM}$ is given $\hat{pk}$ (but not the corresponding $\hat{sk}$), and needs to produce a pair of messages whose ciphertexts she can distinguish. $\mathcal{A}_{GM}$ accomplishes this by simulating a channel experiment $\mathbf{Exp}_{g\mathcal{IRC},\mathcal{A}}^{channel\text{–}ind\text{–}b}(k)$ for $\mathcal{A}$ as follows.

First, $\mathcal{A}_{GM}$ guesses a time period $\tau \in [0, q_{refr}-1]$, which she hopes will coincide with $\mathcal{A}$'s challenge period in the simulated experiment. $\mathcal{A}_{GM}$ succeeds only if this guess is correct and if $\mathcal{A}$ mounts a *fro*-attack in this period.

$\mathcal{A}_{GM}$ simulates the channel experiment for $\mathcal{A}$ up to the period τ with perfect fidelity by using $g\mathcal{IRC}.GenR$ to generate all channel secrets and appropriate refresh messages. Responses to $\mathcal{A}$'s oracle queries for these periods are constructed accordingly.

Period τ is initiated with the refresh $R_{\tau-1}^{[\bar{w}]}$ requested by $\mathcal{A}$. Assuming a fro-attack in period τ, the originator w of this refresh will also be the challenge receiver. $\mathcal{A}_{GM}$ constructs the refresh $R_{\tau-1}^{[\bar{w}]}$ by setting the data public key $SK_{\tau}^{[\bar{w}]}.pk^{[w|d\rangle} = \hat{pk}$ and generating the rest of $R_{\tau-1}^{[\bar{w}]}$ as before. Note this does not require knowledge of $\hat{sk}$, and $\mathcal{A}_{GM}$ can still answer $\mathcal{A}$'s *Odec* queries for period τ by querying its own decryption oracle. For all subsequent periods $\mathcal{A}_{GM}$ simulates the experiment for $\mathcal{A}$ as prior to the challenge period: by generating all the keys herself.

Eventually, in order to succeed in the simulated experiment, $\mathcal{A}$ must output two messages m_0, m_1 along with the challenge receiver w', challenge sender $\bar{w}'$, and the challenge period τ', such that the challenge receiver is not compromised in τ'. If $\tau' = \tau$ and $w' = w$, then $\mathcal{A}_{GM}$ outputs the same messages m_0, m_1 and receives back $c = g\mathcal{IRC}.EncM_{SK_{\tau}^{[\bar{w}]}}(m_b) = \mathcal{S}.\mathcal{E}_{\hat{pk}}(m_b)$ for unknown $b \leftarrow_R \{0,1\}$. $\mathcal{A}_{GM}$ passes c to $\mathcal{A}$, which returns its guess $\hat{b}$; $\mathcal{A}_{GM}$ in turn outputs $\hat{b}$ as her own.

If $\tau' \neq \tau$ or $w' \neq w$, or $\mathcal{A}$ exposes $SK_{\tau}^{[w]}$, then $\mathcal{A}_{GM}$ aborts the simulation and outputs a random bit. Since $\mathcal{A}_{GM}$ generates, and thus knows, all the channel secrets except for $SK_{\tau}^{[w]}.sk^{[w|d\rangle}$, she can simulate all the interactions with $\mathcal{A}$, except for the abortive case of exposing $SK_{\tau}^{[w]}$ (which requires no simulation).

Clearly, for all executions in which $\mathcal{A}_{GM}$ correctly guesses the challenge period (with probability $1/q_{refr}$) $\mathcal{A}_{GM}$'s advantage will be $\epsilon_{fro}(k)$. Thus the overall advantage of $\mathcal{A}_{GM}$ is

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}}^{ind\text{-}cca}(k) = \frac{\epsilon_{fro}(k)}{q_{refr}}.$$

The running time characteristics of $\mathcal{A}_{GM}$ are straightforward from the above. $\square$

C.3 The To-Attack Case: Public-Key Encryption Chains and Their Security.

Consider the following extension to the security experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{GM}}^{ind\text{-}cca\text{-}b}$ (Sec. A.1), in which the CCA-adversary is now allowed to interact with successive elements of a "chain" consisting of pairs of instances of $\mathcal{S}$. Specifically, upon each "next" request to see a new element i in the chain, the adversary receives (i) the new pair of public keys $(pk_i^{[d\rangle}, pk_i^{[\rho\rangle})$ (data and refresh), and (ii) an encryption c_i (under the refresh public key of the previous instance) of a plaintext consisting of the new corresponding pair of secret keys $(sk_i^{[d\rangle}, sk_i^{[\rho\rangle})$ concatenated with an arbitrary suffix X_i selected by $\mathcal{A}_{Chain}$. That is, $c_i = \mathcal{E}_{pk_{i-1}^{[\rho\rangle}}(sk_i^{[d\rangle}, sk_i^{[\rho\rangle}, X_i)$[18]. Eventually $\mathcal{A}_{Chain}$ selects one element τ in the chain against whose data secret key it attempts to distinguish ciphertexts.

Note that the adversary $\mathcal{A}_{Chain}$ retains CCA powers against every *data* stream instance of $\mathcal{S}$ in the chain: if t is the current chain element, then for any $\tau \leq t$ and any y not equal to the ciphertext challenge, a query of (τ, y) to $\mathcal{A}_{Chain}$'s $Odec_t$ oracle returns $\mathcal{D}_{sk_{\tau}^{[d\rangle}}(y)$.

Let $\mathcal{A}_{Chain}$ be an adversary, and let $\mathcal{S}$, b and k be as in Section A.1. We use s_{i+1} to denote the state of $\mathcal{A}_{Chain}$ after computing on element i in the chain. The chain experiment in detail is given by:

Experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind\text{-}b}(k)$

```
$t \leftarrow 0$; $c \leftarrow \perp$; $c_0 \leftarrow \perp$; $s_0 \leftarrow \perp$; $q \leftarrow$ "next"
$(sk_t^{[d\rangle}, pk_t^{[d\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial data key pair
$(sk_t^{[\rho\rangle}, pk_t^{[\rho\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial refresh key pair
while $q ==$ "next" do :
    $(q, X_{t+1}, s_{t+1}) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(pk_t^{[d\rangle}, pk_t^{[\rho\rangle}, c_t, s_t)$ % next request
    CASE $q$
        "next":
            $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$ % next data key pair
            $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$ % next refresh key pair
            $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(sk_{t+1}^{[d\rangle}, sk_{t+1}^{[\rho\rangle}, X_{t+1})$ % ciphertext of new secret keys plus suffix
            $t \leftarrow t + 1$
        "test":
            if $c \neq \perp$ then break % only one challenge allowed
```

[18] Note that this ciphertext format matches that of a refresh message in the $g\mathcal{IRC}$ construction (Section 3); hence the adversary $\mathcal{A}_{Chain}$, by selecting a proper suffix X_i and querying for the next chain element, will be able to obtain a good-looking crucial to-refresh even without knowing the prefix of its plaintext.

```
        $(m_0, m_1, \tau) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(s_{t+1})$  % request for ciphertext challenge for some instance $\tau \in$
            $[0, t]$
        $c \leftarrow \mathcal{S}.\mathcal{E}_{pk_\tau^{[d\rangle}}(m_b)$  % ciphertext challenge
        $\hat{b} \leftarrow \mathcal{A}_{Chain}^{Odec_t}(c, s_{t+1})$  % guess $b$
    return $(\tau, \hat{b})$
```

We say that $\mathcal{A}_{Chain}$ as above is a *chain $\mathcal{S}$-adversary* and define the *chain advantage* of $\mathcal{A}_{Chain}$ against $\mathcal{S}$ as:

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{--}ind}(k) = Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{--}ind\text{--}1}(k) : \hat{b} = 1] - Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{--}ind\text{--}0}(k) : \hat{b} = 1].$$

C.4 The To-Attack Case: Proof of Lemma 7

Proof. Let chain $\mathcal{S}$-adversary $\mathcal{A}_{Chain}$ attack an $\mathcal{S}$-chain consisting of data keys $(\widehat{sk_i^{[d\rangle}}, \widehat{pk_i^{[d\rangle}})$, refresh keys $(\widehat{sk_i^{[\rho\rangle}}, \widehat{pk_i^{[\rho\rangle}})$, and encryptions of secret key pairs $c_i = \mathcal{S}.\mathcal{E}_{\widehat{pk_{i-1}^{[\rho\rangle}}}\left(\widehat{sk_i^{[d\rangle}}, \widehat{sk_i^{[\rho\rangle}}, X_i\right)$, for $i = 1, \ldots, q_{refr} - 1$. Recall that in period i $\mathcal{A}_{Chain}$ is given $\widehat{pk_i^{[d\rangle}}$, $\widehat{pk_i^{[\rho\rangle}}$, and c_i, and its goal is to output a pair of messages m_0, m_1 along with a *chain* challenge period λ such that it can distinguish between their encryptions under the chain data key for chain period λ.

First, $\mathcal{A}_{Chain}$ guesses a *channel* challenge period $\tau' \in [0, q_{refr} - 1]$, a period $\sigma' \in [0, \tau']$, and a challenge receiver $u' \in \{0, 1\}$.

Next, $\mathcal{A}_{Chain}$ simulates the channel experiment for $\mathcal{A}$ up through period $\sigma' - 1$ with perfect fidelity by using $g\mathcal{IRC}.GenR$ to generate all channel secrets and appropriate refresh messages. Responses to $\mathcal{A}$'s oracle queries for these periods are constructed accordingly.

Period σ' is initiated with the refresh $R_{\sigma'-1}^{[]}$ requested by $\mathcal{A}$. Assuming that all of the above guesses of $\mathcal{A}_{Chain}$ lead to a *useful* simulation (defined below), then this refresh will either be a fro-refresh or a non Q_r-queried to-refresh with respect to the challenge receiver u'. Knowing this, $\mathcal{A}_{Chain}$ uses period σ' as the "graft point" for beginning the insertion of the *chain* keys into the channel simulation, which it continues up through the challenge period τ'. Assume without loss of generality that $u' = 0$. Then, for all periods $t \in [\sigma', \tau']$, $\mathcal{A}_{Chain}$ sets $SK_t^{[1]}.pk^{[0|d\rangle} = \widehat{pk_{t-\sigma'}^{[d\rangle}}$ and $SK_t^{[1]}.pk^{[0|\rho\rangle} = \widehat{pk_{t-\sigma'}^{[\rho\rangle}}$, while $SK_t^{[0]}.sk^{[0|d\rangle} = \widehat{sk_{t-\sigma'}^{[d\rangle}}$ and $SK_t^{[0]}.sk^{[0|\rho\rangle} = \widehat{sk_{t-\sigma'}^{[\rho\rangle}}$ are unknown. $\mathcal{A}_{Chain}$ generates the other half of these keys as before (using $g\mathcal{IRC}.GenR$). In order to obtain the to-refresh messages initiating the channel periods $t \in [\sigma' + 1, \tau']$, $\mathcal{A}_{Chain}$ sets suffix $X_{t-\sigma'} = \left(SK_t^{[0]}.pk^{[1|d\rangle}, SK_t^{[0]}.pk^{[1|\rho\rangle}\right)$ and queries for the next chain element. This yields an encryption $c_{t-\sigma'} = \mathcal{E}_{\widehat{pk_{t-\sigma'-1}^{[\rho\rangle}}}\left(\widehat{sk_{t-\sigma'}^{[d\rangle}}, \widehat{sk_{t-\sigma'}^{[\rho\rangle}}, X_{t-\sigma'}\right)$, which $\mathcal{A}_{Chain}$ supplies upon request to $\mathcal{A}$ as to-refresh message $R_{t-1}^{[u']}$. Note that $\mathcal{A}_{Chain}$ can still answer all of $\mathcal{A}$'s *Odec* oracle queries: for one direction of the channel $\mathcal{A}_{Chain}$ continues to generate all keys, and for the other direction $\mathcal{A}_{Chain}$ can query its own *Odec* oracle for the appropriate chain period. Once the challenge period τ' passes, $\mathcal{A}_{Chain}$ returns to simulating the experiment for $\mathcal{A}$ as prior to period σ'.

Eventually, in order to succeed in the simulated experiment, $\mathcal{A}$ must output a challenge message pair, m_0 and m_1, along with the challenge receiver u, challenge sender $\bar{u}$, and the challenge period τ, such that the challenge receiver is not compromised in period τ. A simulation is *useful* if $\tau' = \tau$, $u' = u$, the refresh $R^{[]}_{\sigma'-1}$ is either a fro-refresh or a non Q_r-queried to-refresh with respect to u', all refreshes initiating periods $\sigma' + 1$ up to and including period τ' are to-refreshes and are all Q_r-queried by $\mathcal{A}$, and $SK^{[u']}_{\tau}$ is not Q-exposed[19]. In a useful simulation, note that Q-exposure of *any* key $SK^{[u']}_{t}$, $\sigma' \leq t \leq \tau'$ combined with the set of Q_r-queried refreshes would result in the Q-exposure of $SK^{[u']}_{\tau'}$ itself via evolution. Thus in a useful simulation $\mathcal{A}_{Chain}$ knows all the secrets necessary to answer $\mathcal{A}$'s queries.

If the simulation is useful, $\mathcal{A}_{Chain}$ outputs the same challenge message pair, m_0 and m_1, as $\mathcal{A}$, along with chain period $\lambda = \tau' - \sigma'$, and receives back $c = g\mathcal{IRC}.EncM_{SK^{[\bar{u}]}_{\tau'}}(m_b) = \mathcal{S}.\mathcal{E}_{\widehat{pk^{[d\rangle}_{\tau'-\sigma'}}}(m_b)$ for some unknown $b \leftarrow_R \{0,1\}$. $\mathcal{A}_{Chain}$ passes c to $\mathcal{A}$, which returns its guess $\hat{b}$; $\mathcal{A}_{Chain}$ in turn outputs $\hat{b}$ as her own.

For all simulations which are not useful, $\mathcal{A}_{Chain}$ aborts the simulation and outputs a random bit.

Clearly, when $\mathcal{A}_{Chain}$ has a useful simulation her advantage will be $\epsilon_{to}(k)$. This occurs when she correctly guesses the challenge receiver, challenge period, and graft point (with probabilities $1/2$, $1/q_{refr}$, and at least $1/q_{refr}$). Thus the overall advantage of $\mathcal{A}_{Chain}$ is

$$\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k) \geq \frac{\epsilon_{to}(k)}{2{q_{refr}}^2}.$$

The running time characteristics of $\mathcal{A}_{Chain}$ are straightforward from the above. □

C.5 The To-Attack Case: Proof of Lemma 8

Proof. Given a chain $\mathcal{S}$-adversary $\mathcal{A}_{Chain}$, we describe below how to construct a pair of ordinary $\mathcal{S}$-adversaries, $\mathcal{A}^{rand}_{GM}$ and $\mathcal{A}^{hyb}_{GM}$, such that at least one of the pair will have ordinary advantage against $\mathcal{S}$ comparable to the chain advantage of $\mathcal{A}_{Chain}$.

The random adversary $\mathcal{A}^{rand}_{GM}$ $\mathcal{A}^{rand}_{GM}$, working inside of $\mathbf{Exp}^{ind\text{-}cca\text{-}b}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}$, uses $\mathcal{A}_{Chain}$ as an oracle to distinguish against a single instance $(\hat{sk}, \hat{pk})$ of $\mathcal{S}$. $\mathcal{A}^{rand}_{GM}$ simulates for $\mathcal{A}_{Chain}$ a modified version of the chain experiment, denoted by $\mathbf{Exp}^{chain\text{-}ind\text{-}b}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1)$, in which upon every request to see the next element in the chain, $\mathcal{A}_{Chain}$ receives a new pair of public keys (data and refresh, as in the original chain experiment) together with an encryption of a plaintext whose format is different from that in the original chain experiment: the data and refresh secret key portions are both replaced by fresh random strings generated by $\mathcal{A}^{rand}_{GM}$. By substituting $\hat{pk}$ for the data public key of a randomly selected element in the chain, $\mathcal{A}^{rand}_{GM}$ is able to leverage whatever advantage $\mathcal{A}_{Chain}$ may retain in this modified chain experiment. More details of how $\mathcal{A}^{rand}_{GM}$ works are given below; see Appendix C.7 for precise pseudocode.

$\mathcal{A}^{rand}_{GM}$ first uniformly selects $\tau' \in [0, q_{next}-1]$ as its guess of the data key instance (time period[20]) of $\mathcal{S}$ against which $\mathcal{A}_{Chain}$ will eventually choose to distinguish. $\mathcal{A}^{rand}_{GM}$ also selects a random "back-

[19] The degenerate case $\sigma' = \tau'$ covers a to-attack in which $\mathcal{A}$ does not expose the crucial refresh.

[20] For convenience we will subsequently refer to the consecutive instances of $\mathcal{S}$ generated within the chain experiment as if they corresponded to consecutive time periods.

up" distinguishing guess $g \leftarrow_R \{0,1\}$ for use in case its guess τ' proves to be incorrect. $\mathcal{A}_{GM}^{rand}$ then simulates $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}b}(k, q_{next}-1)$ for $\mathcal{A}_{Chain}$. As in the original chain experiment, for all time periods $1 \leq i \leq q_{next}-1$ (i.e. the entire chain[21]), $\mathcal{A}_{Chain}$ receives the new pair of public keys (data and refresh): $(pk_i^{[d\rangle}, pk_i^{[\rho\rangle})$, but the corresponding ciphertext c_i is now constructed as: $c_i = \mathcal{E}_{pk_{i-1}^{[\rho\rangle}}(R_i^{[d\rangle}, R_i^{[\rho\rangle}, X_i)$. Specifically, in the plaintext corresponding to c_i the data secret key $sk_i^{[d\rangle}$ and refresh secret key $sk_i^{[\rho\rangle}$ are replaced by the fresh random strings $R_i^{[d\rangle}$ and $R_i^{[\rho\rangle}$ generated by $\mathcal{A}_{GM}^{rand}$ and of equal length as the secret keys. Furthermore, $\mathcal{A}_{GM}^{rand}$ makes the *data* secret key substitution $pk_{\tau'}^{[d\rangle} \leftarrow \hat{pk}$. Note that $\mathcal{A}_{GM}^{rand}$ can answer all of $\mathcal{A}_{Chain}$'s *Odec* queries, either by using the appropriate data decryption secret (for all periods $i \neq \tau'$) or by querying its own *Odec* oracle (for $i = \tau'$).

When $\mathcal{A}_{GM}^{rand}$ observes $\mathcal{A}_{Chain}$'s ciphertext challenge request (m_0, m_1, τ), it queries its own oracle to obtain the ciphertext challenge $c = \mathcal{S}.\mathcal{E}_{\hat{pk}}(m_b)$, which it hands to $\mathcal{A}_{Chain}$. $\mathcal{A}_{GM}^{rand}$ waits until $\mathcal{A}_{Chain}$ outputs its distinguishing guess $(\tau, \hat{b})$ to check whether $\tau = \tau'$. If equality holds (i.e. $\mathcal{A}_{GM}^{rand}$ correctly guessed the distinguishing time period of $\mathcal{A}_{Chain}$), then $\mathcal{A}_{GM}^{rand}$ outputs the $\hat{b}$ supplied by $\mathcal{A}_{Chain}$ as its own distinguishing guess. Otherwise, if $\tau \neq \tau'$, then $\mathcal{A}_{GM}^{rand}$ outputs its back-up guess g.

Let $\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind}(k, q_{next}-1)$ denote the advantage of $\mathcal{A}_{Chain}$ in the modified chain experiment. Then it is straightforward to compute the advantage of $\mathcal{A}_{GM}^{rand}$ as:

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{–}cca}(k) = \frac{1}{q_{next}} \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind}(k, q_{next}-1).$$

See Appendix C.8 for details of the calculation. The running time characteristics of $\mathcal{A}_{GM}^{rand}$ are straightforward from the above.

The hybrid adversary $\boldsymbol{\mathcal{A}_{GM}^{hyb}}$ $\mathcal{A}_{GM}^{hyb}$, working inside of $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca\text{–}b}$, likewise uses $\mathcal{A}_{Chain}$ as an oracle to distinguish against a single instance $(\hat{sk}, \hat{pk})$ of $\mathcal{S}$. However, in order to leverage the potential *difference* between the advantages of $\mathcal{A}_{Chain}$ in experiments $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}b}(k)$ and $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}b}(k, q_{next}-1)$, $\mathcal{A}_{GM}^{hyb}$'s strategy is "hybrid" in nature, complimenting that of $\mathcal{A}_{GM}^{rand}$ above.

$\mathcal{A}_{GM}^{hyb}$ first selects a random index $j \leftarrow_R [0, q_{next}–2]$[22] and a bit $d \leftarrow_R \{0,1\}$, and then simulates for $\mathcal{A}_{Chain}$ a modified version of the chain experiment, denoted $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, j+b)$. This experiment is a hybrid of the original chain experiment and the modified version simulated by $\mathcal{A}_{GM}^{rand}$ described above. More precisely, as in the original chain experiment $\mathcal{A}_{Chain}$ receives for all chain periods the new pair of public keys (data and refresh): $(pk_i^{[d\rangle}, pk_i^{[\rho\rangle})$, but the parameter $j+b$ specifies that for all time periods $1 \leq i \leq j+b$ the corresponding ciphertext c_i is constructed as in the simulation of $\mathcal{A}_{GM}^{rand}$: $c_i = \mathcal{E}_{pk_{i-1}^{[\rho\rangle}}(R_i^{[d\rangle}, R_i^{[\rho\rangle}, X_i)$. That is, for the first $j+b$ time periods in this hybrid experiment[23], in the plaintext corresponding to c_i the data secret key $sk_i^{[d\rangle}$ and refresh secret key $sk_i^{[\rho\rangle}$ are replaced by the fresh random strings $R_i^{[d\rangle}$ and $R_i^{[\rho\rangle}$ generated by $\mathcal{A}_{GM}^{hyb}$ and of equal length as the secret keys.

$\mathcal{A}_{GM}^{hyb}$ accomplishes this hybrid simulation with index $j+b$ (where bit b is unknown to $\mathcal{A}_{GM}^{hyb}$) as follows. $\mathcal{A}_{GM}^{hyb}$ makes the *refresh* secret key substitution $pk_j^{[\rho\rangle} \leftarrow \hat{pk}$. Then, when it comes time

[21] Not counting period 0, for which $\mathcal{A}_{Chain}$ receives no ciphertext at all.
[22] Note if the bound $q_{next} < 2$ then $\mathcal{A}_{Chain}$ reduces immediately to an ordinary $\mathcal{S}$-adversary, so we assume $q_{next} \geq 2$.
[23] Not counting period 0, for which $\mathcal{A}_{Chain}$ receives no ciphertext at all.

to construct the ciphertext c_{j+1} to be handed to $\mathcal{A}_{Chain}$ along with the new public keys $pk_{j+1}^{[d\rangle}$ and $pk_{j+1}^{[\rho\rangle}$, $\mathcal{A}_{GM}^{hyb}$ queries its own ciphertext challenge oracle on the message pair

$$\left(M_0 = (sk_{j+1}^{[d\rangle}, sk_{j+1}^{[\rho\rangle}, X_{j+1}), M_1 = (R_{j+1}^{[d\rangle}, R_{j+1}^{[\rho\rangle}, X_{j+1})\right),$$

obtaining ciphertext C. $\mathcal{A}_{GM}^{hyb}$ then substitutes this as the chain ciphertext: $c_{j+1} \leftarrow C$.

Then, for all $i > j+1$, the simulation for $\mathcal{A}_{Chain}$ is identical to the original chain experiment: for each time period $i > j+1$, $\mathcal{A}_{Chain}$ again receives $pk_i^{[d\rangle}$, $pk_i^{[\rho\rangle}$ and $c_i = \mathcal{E}_{pk_{i-1}^{[\rho\rangle}}(sk_i^{[d\rangle}, sk_i^{[\rho\rangle}, X_i)$.

Furthermore, using the bit d of its own choosing, $\mathcal{A}_{GM}^{hyb}$ constructs the ciphertext challenge on which $\mathcal{A}_{Chain}$ attempts to distinguish. When $\mathcal{A}_{GM}^{hyb}$ observes the distinguishing guess $(\tau, \hat{d})$ output by $\mathcal{A}_{Chain}$, it checks to see whether or not $\hat{d} = d$ (i.e. whether or not $\mathcal{A}_{Chain}$ distinguished successfully). If equality holds, then $\mathcal{A}_{GM}^{hyb}$ outputs $\hat{b} \leftarrow 0$ as its distinguishing guess. Otherwise, if $\hat{d} \neq d$, then $\mathcal{A}_{GM}^{hyb}$ outputs $\hat{b} \leftarrow 1$. Recall that if $b = 0$ $\mathcal{A}_{Chain}$ will see one fewer ciphertexts of random string pairs in the simulation than if $b = 1$. Thus $\mathcal{A}_{GM}^{hyb}$'s strategy is an attempt to detect any cumulative decay in the advantage of $\mathcal{A}_{Chain}$ resulting from the substitution of encryptions of randomness. See Appendix C.9 for precise pseudocode.

It is straightforward to compute the advantage of $\mathcal{A}_{GM}^{hyb}$ as:

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{-}cca}(k) = \frac{1}{2(q_{next}-1)}\left(\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind}(k) - \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind}(k, q_{next}-1)\right).$$

See Appendix C.10 for details of the calculation. The running time characteristics of $\mathcal{A}_{GM}^{hyb}$ are straightforward from the above.

Maximum advantage of the pair $(\mathcal{A}_{GM}^{rand}, \mathcal{A}_{GM}^{hyb})$ From the bounds given above for the advantage of $\mathcal{A}_{GM}^{rand}$ and $\mathcal{A}_{GM}^{hyb}$, it is straightforward algebra to show that we will always have

$$\max\left\{\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{-}cca}(k),\ \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{-}cca}(k)\right\} \geq \left(\frac{1}{3q_{next}-2}\right)\cdot \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind}(k).$$

See Appendix C.11 for details. □

C.6 The To-Attack Case: Proof of Corollary 1

Proof. Indeed, by Lemma 7, there exists an adversary $\mathcal{A}_{Chain}$ with chain advantage at least $\frac{\epsilon_{to}(k)}{2{q_{refr}}^2}$ against $\mathcal{S}$ and which obtains at most q_{refr} chain elements. Consequently, by Lemma 8, there exists a pair of adversaries $(\mathcal{A}_{GM}^{rand}, \mathcal{A}_{GM}^{hyb})$ such that $\max\{\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{-}cca}(k), \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{-}cca}(k)\} \geq \left(\frac{1}{3q_{refr}-2}\right)\cdot \mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind}(k) \geq \left(\frac{1}{3q_{refr}-2}\right)\cdot\frac{\epsilon_{to}(k)}{2{q_{refr}}^2} = \frac{\epsilon_{to}(k)}{2{q_{refr}}^2(3q_{refr}-2)}$. □

C.7 Supplemental Details: Description of $\mathcal{A}_{GM}^{rand}$

Recall that $\mathcal{A}_{GM}^{rand}$ is an $\mathcal{S}$-adversary attacking an instance with keys $(\hat{sk}, \hat{pk})$. Thus $\mathcal{A}_{GM}^{rand}$ works inside of $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{GM}^{rand}}^{ind\text{-}cca\text{-}b}(k)$ and seeks to distinguish the bit b. $\mathcal{A}_{GM}^{rand}$ uses $\mathcal{A}_{Chain}$ as an oracle.

$\tau' \leftarrow_R [0, q_{next}-1]$; $g \leftarrow_R \{0,1\}$ % $\mathcal{A}_{GM}^{rand}$ guesses chain challenge period τ' and prepares "back-up" distinguishing guess g

Experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{-}ind\text{-}b}(k, q_{next}-1)$ % as simulated for $\mathcal{A}_{Chain}$ by $\mathcal{A}_{GM}^{rand}$

```
$t \leftarrow 0$; $c \leftarrow \perp$; $c_0 \leftarrow \perp$; $s_0 \leftarrow \perp$; $q \leftarrow$ "next"
$(sk_t^{[d\rangle}, pk_t^{[d\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial data key pair
$(sk_t^{[\rho\rangle}, pk_t^{[\rho\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial refresh key pair
if $t = \tau'$ then $pk_t^{[d\rangle} \leftarrow \hat{pk}$ % check for base case substitution of data public key
while $q ==$ "next" do :
    $(q, X_{t+1}, s_{t+1}) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(pk_t^{[d\rangle}, pk_t^{[\rho\rangle}, c_t, s_t)$ % next request
        CASE $q$
            "next":
                if $t \neq \tau' - 1$ then do: % encrypt randomness
                    $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$ % next data key pair
                    $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$ % next refresh key pair
                    $R_{t+1}^{[d\rangle} \leftarrow_R \{0,1\}^{l(k)}$ % where $l(k) = |sk|$
                    $R_{t+1}^{[\rho\rangle} \leftarrow_R \{0,1\}^{l(k)}$
                    $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(R_{t+1}^{[d\rangle}, R_{t+1}^{[\rho\rangle}, X_{t+1})$ % ciphertext of random strings plus suffix
                    $t \leftarrow t+1$
                else if $t = \tau' - 1$ then do: % encrypt randomness; substitute data public key
                    $pk_{t+1}^{[d\rangle} \leftarrow \hat{pk}$ % substitute data public key
                    $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$ % next refresh key pair
                    $R_{t+1}^{[d\rangle} \leftarrow_R \{0,1\}^{l(k)}$ % where $l(k) = |sk|$
                    $R_{t+1}^{[\rho\rangle} \leftarrow_R \{0,1\}^{l(k)}$
                    $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(R_{t+1}^{[d\rangle}, R_{t+1}^{[\rho\rangle}, X_{t+1})$ % ciphertext of random strings plus suffix
                    $t \leftarrow t+1$

            "test":
                if $c \neq \perp$ then break % only one challenge allowed
                let $(m_0, m_1, \tau) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(s_{t+1})$ % request for ciphertext challenge for some instance $\tau \in [0, t]$
                $c \leftarrow \mathcal{S}.\mathcal{E}_{pk_\tau^{[d\rangle}}(m_b)$ % $\mathcal{A}_{GM}^{rand}$ obtains ciphertext challenge $c$ from its oracle
                $\hat{b} \leftarrow \mathcal{A}_{Chain}^{Odec_t}(c, s_{t+1})$ % guess $b$
    return $(\tau, \hat{b})$ % observed by $\mathcal{A}_{GM}^{rand}$
```

Once $\mathcal{A}_{GM}^{rand}$ observes the output $(\tau, \hat{b})$ of $\mathcal{A}_{Chain}$, it checks to see whether $\tau' = \tau$. If equality holds, then $\mathcal{A}_{GM}^{rand}$ outputs $\hat{b}$ as its own distinguishing guess. Otherwise, if $\tau' \neq \tau$, $\mathcal{A}_{GM}^{rand}$ outputs its back-up guess g.

C.8 Supplemental Details: Calculating Advantage of $\mathcal{A}_{GM}^{rand}$

We calculate the advantage of $\mathcal{A}_{GM}^{rand}$ against $\mathcal{S}$.

By definition (A.1) we have:

$$\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k) = Pr[\mathbf{Exp}^{ind\text{–}cca\text{–}1}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k) = 1] - Pr[\mathbf{Exp}^{ind\text{–}cca\text{–}0}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k) = 1]. \tag{3}$$

According to the above specification of $\mathcal{A}^{rand}_{GM}$, the LHS of this difference (3) is the same as:

$$Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1); \tau' \leftarrow [0, q_{next} - 1]; g \leftarrow [0, 1] : \tau = \tau' \wedge \hat{b} = 1]$$

$$+Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1); \tau' \leftarrow [0, q_{next} - 1]; g \leftarrow [0, 1] : \tau \neq \tau' \wedge g = 1]$$

Expanding over the independent parameter τ' and simplifying gives:

$$= \frac{1}{q_{next}} \sum_{\tau'=0}^{q_{next}-1} Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1); g \leftarrow [0, 1] : \tau = \tau' \wedge \hat{b} = 1]$$

$$+\frac{1}{q_{next}} \sum_{\tau'=0}^{q_{next}-1} Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1); g \leftarrow [0, 1] : \tau \neq \tau' \wedge g = 1]$$

$$= \frac{1}{q_{next}} Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1) : \hat{b} = 1] + \frac{q_{next} - 1}{2q_{next}}.$$

Likewise we find that the RHS of the same difference (3) from above is equivalent to:

$$\frac{1}{q_{next}} Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}0}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1) : \hat{b} = 1] + \frac{q_{next} - 1}{2q_{next}}.$$

Thus we may write the original advantage as:

$$\begin{aligned}\mathbf{Adv}^{ind\text{–}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k) &= \frac{1}{q_{next}} \Big(Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}1}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1) : \hat{b} = 1] \\ &\quad - Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}^{chain\text{–}ind\text{–}0}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1) : \hat{b} = 1] \Big) \\ &= \frac{1}{q_{next}} \mathbf{Adv}^{chain\text{–}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next} - 1).\end{aligned}$$

C.9 Supplemental Details: Description of $\mathcal{A}^{hyb}_{GM}$

The precise rules of this experiment are given below:

Recall that $\mathcal{A}^{hyb}_{GM}$ is an $\mathcal{S}$-adversary attacking an instance with keys $(\hat{sk}, \hat{pk})$. Thus $\mathcal{A}^{hyb}_{GM}$ works inside of $\mathbf{Exp}^{ind\text{–}cca\text{–}b}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k)$ and seeks to distinguish the bit b. $\mathcal{A}^{hyb}_{GM}$ uses $\mathcal{A}_{Chain}$ as an oracle.

$j \leftarrow_R [0, q_{next} - 2]$; $d \leftarrow_R \{0, 1\}$ % $\mathcal{A}^{hyb}_{GM}$ picks index j and bit d for constructing $\mathcal{A}_{Chain}$'s eventual ciphertext challenge. Note in the simulation below that $\mathcal{A}^{hyb}_{GM}$ knows the bit d and is trying to distinguish the bit b. $\mathcal{A}^{hyb}_{GM}$ uses $\mathcal{A}_{Chain}$ as an oracle.

Experiment $\mathbf{Exp}^{chain\text{–}ind\text{–}d}_{\mathcal{S},\mathcal{A}_{Chain}}(k, j + b)$ % as simulated for $\mathcal{A}_{Chain}$ by $\mathcal{A}^{hyb}_{GM}$

$t \leftarrow 0$; $c \leftarrow \perp$; $c_0 \leftarrow \perp$; $s_0 \leftarrow \perp$; $q \leftarrow$ "next"

```
$(sk_t^{[d\rangle}, pk_t^{[d\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial data key pair
$(sk_t^{[\rho\rangle}, pk_t^{[\rho\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial refresh key pair
if $t = j$ then $pk_t \leftarrow \hat{pk}$  % check for base case substitution of refresh public key
while $q ==$ “next” do :
    $(q, X_{t+1}, s_{t+1}) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(pk_t^{[d\rangle}, pk_t^{[\rho\rangle}, c_t, s_t)$  % next request
        CASE $q$
            “next”:
                if $t < j-1$ then do:  % encrypt randomness
                    $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$  % next data key pair
                    $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$  % next refresh key pair
                    $R_{t+1}^{[d\rangle} \leftarrow_R \{0,1\}^{l(k)}$ % where $l(k) = |sk|$
                    $R_{t+1}^{[\rho\rangle} \leftarrow_R \{0,1\}^{l(k)}$
                    $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(R_{t+1}^{[d\rangle}, R_{t+1}^{[\rho\rangle}, X_{t+1})$  % ciphertext of random strings plus suffix
                    $t \leftarrow t+1$
                else if $t = j-1$ then do:  % encrypt randomness; substitute refresh public key
                    $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$  % next data key pair
                    $pk_{t+1}^{[\rho\rangle} \leftarrow \hat{pk}$ % substitute refresh public key
                    $R_{t+1}^{[d\rangle} \leftarrow_R \{0,1\}^{l(k)}$ % where $l(k) = |sk|$
                    $R_{t+1}^{[\rho\rangle} \leftarrow_R \{0,1\}^{l(k)}$
                    $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(R_{t+1}^{[d\rangle}, R_{t+1}^{[\rho\rangle}, X_{t+1})$  % ciphertext of random strings plus suffix
                    $t \leftarrow t+1$
                else if $t = j$ then do:  % substitute ciphertext challenge (secret keys?/randomness?)
                    $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$  % next data key pair
                    $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$  % next refresh key pair
                    $R_{t+1}^{[d\rangle} \leftarrow_R \{0,1\}^{l(k)}$ % where $l(k) = |sk|$
                    $R_{t+1}^{[\rho\rangle} \leftarrow_R \{0,1\}^{l(k)}$
                    $(M_0, M_1) \leftarrow \Big((sk_{t+1}^{[d\rangle}, sk_{t+1}^{[\rho\rangle}, X_{t+1}), (R_{t+1}^{[d\rangle}, R_{t+1}^{[\rho\rangle}, X_{t+1})\Big)$ % challenge message pair
                    $C \leftarrow \mathcal{S}.\mathcal{E}_{\hat{pk}}(M_b)$  % $\mathcal{A}_{GM}^{hyb}$ obtains ciphertext challenge $C$ from its oracle
                    $c_{t+1} \leftarrow C$  % substitute challenge
                    $t \leftarrow t+1$
                else if $t > j$ then do:  % encrypt secret keys
                    $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$  % next data key pair
                    $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$  % next refresh key pair
                    $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(sk_{t+1}^{[d\rangle}, sk_{t+1}^{[\rho\rangle}, X_{t+1})$  % ciphertext of new secret keys plus suffix
                    $t \leftarrow t+1$

                “test”:
                    if $c \neq \perp$ then break  % only one challenge allowed
                    let $(m_0, m_1, \tau) \leftarrow \mathcal{A}_{Chain}^{Odec_t}(s_{t+1})$  % request for ciphertext challenge for some
                     instance $\tau \in [0, t]$
                    $c \leftarrow \mathcal{S}.\mathcal{E}_{pk_\tau^{[d\rangle}}(m_d)$  % $\mathcal{A}_{GM}^{hyb}$ generates ciphertext challenge $c$
        $\hat{d} \leftarrow \mathcal{A}_{Chain}^{Odec_t}(c, s_{t+1})$  % guess $d$
    return $(\tau, \hat{d})$ % observed by $\mathcal{A}_{GM}^{hyb}$
```

Once $\mathcal{A}_{GM}^{hyb}$ observes the output $(\tau, \hat{d})$ of $\mathcal{A}_{Chain}$, it checks to see whether $\hat{d} = d$ (i.e. whether or not $\mathcal{A}_{Chain}$ distinguished correctly). If equality holds, then $\mathcal{A}_{GM}^{hyb}$ outputs $\hat{b} \leftarrow 0$ as its distinguishing guess. Otherwise, if $\hat{d} \neq d$, $\mathcal{A}_{GM}^{hyb}$ outputs $\hat{b} \leftarrow 1$.

C.10 Supplemental Details: Calculating Advantage of $\mathcal{A}_{GM}^{hyb}$

Generalizing previous notation, let $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k,l)$ denote the modified chain experiment in which $\mathcal{A}_{Chain}$ receives encryptions of l fresh random values $R_1, \ldots, R_l$ in place of encryptions of the first l secret keys $sk_1, \ldots, sk_l$ in the chain[24]. For $l = 0$ this reduces to the original chain experiment: $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k,0) = \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k)$. Likewise let the advantage of $\mathcal{A}_{Chain}$ in this modified experiment be denoted as $\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind}(k,l)$.
We now calculate the advantage of $\mathcal{A}_{GM}^{hyb}$ against $\mathcal{S}$. By definition (A.1) we have:

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca}(k) = Pr[\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca\text{–}1}(k) = 1] - Pr[\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{GM}^{hyb}}^{ind\text{–}cca\text{–}0}(k) = 1]. \tag{4}$$

By the above specification of $\mathcal{A}_{GM}^{hyb}$ this difference (4) is the same as:

$$Pr[(\tau,\hat{d}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, j+1); j \leftarrow [0, q_{next}-2]; d \leftarrow [0,1] : \hat{d} \neq d]$$

$$-Pr[(\tau,\hat{d}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, j); j \leftarrow [0, q_{next}-2]; d \leftarrow [0,1] : \hat{d} \neq d].$$

Expanding over the independent parameters j and d gives:

$$= \frac{1}{q_{next}-1} \sum_{j=0}^{q_{next}-2} \frac{1}{2} \sum_{d=0}^{1} Pr[(\tau,\hat{d}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, j+1) : \hat{d} \neq d] \tag{5}$$

$$- \frac{1}{q_{next}-1} \sum_{j=0}^{q_{next}-2} \frac{1}{2} \sum_{d=0}^{1} Pr[(\tau,\hat{d}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, j) : \hat{d} \neq d]. \tag{6}$$

To simplify this expression, let Y_l denote the chain adversary's probability of correct guessing in the experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k,l)$ for $d \leftarrow_R \{0,1\}$:

$$Y_l \overset{def}{=} \frac{1}{2} \sum_{d=0}^{1} Pr[(\tau,\hat{d}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{Chain}}^{chain\text{–}ind\text{–}d}(k, l) : \hat{d} = d].$$

Using this notation we may rewrite the above (5) and (6) as

$$= \frac{1}{q_{next}-1} \sum_{j=0}^{q_{next}-2} (1 - Y_{j+1}) - \frac{1}{q_{next}-1} \sum_{j=0}^{q_{next}-2} (1 - Y_j) = \frac{1}{q_{next}-1}(Y_0 - Y_{(q_{next}-1)}).$$

[24] Not counting period 0, for which $\mathcal{A}_{Chain}$ receives no ciphertext at all.

Thus for the advantage we have:

$$\begin{aligned}\mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k) &= \frac{1}{q_{next}-1}(Y_0 - Y_{(q_{next}-1)}) \\ &= \frac{1}{2(q_{next}-1)}\left(\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k,0) - \mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next}-1)\right) ,\end{aligned}$$

where the last equality follows from the relation between the probability of correct guessing and advantage: $\forall l,\ Y_l = \frac{1}{2} + \frac{\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k,l)}{2}$. Finally, since by definition

$$\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k,0) = \mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k) ,$$

the result follows.

C.11 Supplemental Details: Maximum advantage of the pair $(\mathcal{A}^{rand}_{GM}, \mathcal{A}^{hyb}_{GM})$

As convenient shorthand let a_0 denote the advantage $\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k,0) = \mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k)$ and let a_{q-1} denote the advantage $\mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k, q_{next}-1)$. Then by the above lower-bounds on the advantages of $\mathcal{A}^{rand}_{GM}$ and $\mathcal{A}^{hyb}_{GM}$ we have:

$$\max\left\{\mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k),\ \mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k)\right\} \geq \max\left\{\frac{a_{q-1}}{q_{next}},\ \frac{a_0 - a_{q-1}}{2(q_{next}-1)}\right\}.$$

Consider the two cases below.

Case I: $a_{q-1} \geq \left(\frac{q_{next}}{3q_{next}-2}\right)\cdot a_0$. This yields

$$\max\left\{\frac{a_{q-1}}{q_{next}},\ \frac{a_0 - a_{q-1}}{2(q_{next}-1)}\right\} \geq \frac{a_{q-1}}{q_{next}} \geq \left(\frac{1}{3q_{next}-2}\right)\cdot a_0.$$

Case II: $a_{q-1} < \left(\frac{q_{next}}{3q_{next}-2}\right)\cdot a_0$. This yields

$$\max\left\{\frac{a_{q-1}}{q_{next}},\ \frac{a_0 - a_{q-1}}{2(q_{next}-1)}\right\} \geq \frac{a_0 - a_{q-1}}{2(q_{next}-1)} > \left(\frac{1}{3q_{next}-2}\right)\cdot a_0.$$

Thus we will always have

$$\max\left\{\mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}^{rand}_{GM}}(k),\ \mathbf{Adv}^{ind\text{-}cca}_{\mathcal{S},\mathcal{A}^{hyb}_{GM}}(k)\right\} \geq \left(\frac{1}{3q_{next}-2}\right)\cdot a_0 = \left(\frac{1}{3q_{next}-2}\right)\cdot \mathbf{Adv}^{chain\text{-}ind}_{\mathcal{S},\mathcal{A}_{Chain}}(k).$$

□

D Channel Security: Proof of Theorem 2 ($g\mathcal{IRC}$ is Spliceable)

Below we argue a series of claims which collectively suffice to prove Theorem 2. These claims connect incremental security notions which we formulate to bridge the gap between the previously studied notion of key indistinguishability for public key encryption schemes and our notion of channel spliceability. First we recall the key indistinguishability security experiment from [BBDP01].

PKE Key Indistinguishability [BBDP01]

```
Experiment $\mathbf{Exp}^{ik\text{-}cca\text{-}b}_{\mathcal{S},\mathcal{A}_{cca}}(k)$
    $I \xleftarrow{R} \mathcal{G}(1^k)$                                    % generate key information common to both key pairs
    $(pk_0, sk_0) \xleftarrow{R} \mathcal{K}(I)$                           % generate two key pairs using common information
    $(pk_1, sk_1) \xleftarrow{R} \mathcal{K}(I)$
    $(m, s) \leftarrow \mathcal{A}_{cca}^{Odec_{sk_0}(\cdot), Odec_{sk_1}(\cdot)}(\texttt{find}, pk_0, pk_1)$   % output challenge message m
    $c \leftarrow \mathcal{E}_{pk_b}(m)$
    $\hat{b} \leftarrow \mathcal{A}_{cca}^{Odec_{sk_0}(\cdot), Odec_{sk_1}(\cdot)}(\texttt{guess}, c, s)$          % guess which key used to encrypt challenge
    return $\hat{b}$
```

Note that $Odec_{sk_0}(\cdot)$ and $Odec_{sk_1}(\cdot)$ on input c return $\perp$.

The encryption scheme $\mathcal{S}$ is said to be $IK - CCA$ secure if any PPT adversary $\mathcal{A}_{cca}$ has only negligible distinguishing advantage on bit b in the above experiment.

Define a multi-challenge version of $IK - CCA$ security by altering the above experiment to allow the adversary to specify multiple challenges. Thus, m and c become the vectors $\vec{m}$ and $\vec{c}$. Again, as is the case in the single challenge experiment, on input any component of $\vec{c}$, $Odec_{sk_0}$ and $Odec_{sk_1}$ return $\perp$.

Claim. Any public-key encryption scheme $\mathcal{S}$ which is $IK - CCA$ secure for a single challenge is also secure for multiple challenges.

Proof. By a straightforward hybrid argument. The single challenge adversary encrypts the first $k - 1$ challenges with pk_0, uses kth challenge as its own, and encrypts all other challenges with pk_1. Loss in advantage factor is just one over the total number of challenge messages.

The following security experiment is used to define an intermediate security property on public-key encryption schemes as another step towards establishing channel spliceability.

Weak Multi-Challenge PKE Key Indistinguishability

```
Oracle $Odec_b(c)$
    if $b = 1$ and $Oenc_{pk_0}$ does not contain $(\cdot, c)$
        then return $Odec_{sk_1}(c)$   % gave alt. key and enc. oracle not used; decrypt using alt. key
    else return $Odec_{sk_0}(c)$   % use usual decryption oracle
```

```
Experiment $\mathbf{Exp}^{ik\text{-}weakM\text{-}cca\text{-}b}_{\mathcal{S},\mathcal{A}_{cca}}(k)$
INIT:
    $I \xleftarrow{R} \mathcal{G}(1^k)$                         % generate key information common to both key pairs
    $(pk_0, sk_0) \xleftarrow{R} \mathcal{K}(I)$                % generate two key pairs using common information
    $(pk_1, sk_1) \xleftarrow{R} \mathcal{K}(I)$
PROBE:
    $s \leftarrow \mathcal{A}_{cca}^{Oenc_{pk_0}(\cdot), Odec_b(\cdot)}(\texttt{find}, pk_b)$
DIST:
    $\hat{b} \leftarrow \mathcal{A}_{cca}^{Oenc_{pk_0}(\cdot), Odec_b(\cdot)}(\texttt{guess}, s)$   % guess whether oracle encryption key same as given public key
    return $\hat{b}$
```

Claim. Any scheme $\mathcal{S}$ which is $IK-CCA$ secure for multiple challenges is also weak $IK-CCA$ secure for multiple challenges.

Proof. Given a weak key distinguisher $\mathcal{A}$, build a strong key distinguisher $\mathcal{A}'$. $\mathcal{A}'$ emulates the experiment for $\mathcal{A}$ using its two public keys and flipping a coin to decide which key to give to $\mathcal{A}$. Queries made to the encryption oracle by $\mathcal{A}$ become the challenge messages for $\mathcal{A}'$. If $\mathcal{A}$ notices no discrepancy between encryptions by the oracle, i.e. $\mathcal{A}'$ challenge ciphertexts, and the encryptions under the key given to it, it outputs 0 and $\mathcal{A}'$ outputs the bit for the key it gave to $\mathcal{A}$. Otherwise, if $\mathcal{A}$ outputs 1, $\mathcal{A}'$ outputs the bit for the key it did not give to $\mathcal{A}$.

The final intermediate public-key encryption security property is given below. As shown below, this is the last step needed in the bridge linking ordinary key indistinguishability to channel spliceability.

Weak PKE Key Indistinguishability Chain

```
Oracle $Odec_{b,t,\tau}(\alpha, c)$
    if $b = 1$ and $\alpha = \tau$ and $Oenc_t$ does not contain $(\alpha, \cdot, c)$
        then return $Odec_{\widehat{sk^{[d\rangle}}}(c)$ % gave alt. key and enc. oracle not used; decrypt using alt. key
    else return $Odec_t(\alpha, c)$ % use usual decryption oracle


Oracle $Oenc_{sk_\tau^{[d\rangle}}(m)$
    if $\tau = \perp$ then return $\perp$ % challenge period $\tau$ still undefined
    else return $\mathcal{E}_{pk_\tau^{[d\rangle}}(m)$


Experiment $\mathbf{Exp}_{\mathcal{S},\mathcal{A}_{KIChain}}^{chain\text{-}weakM\text{-}ik\text{-}cca\text{-}b}(k)$
INIT:
    $t \leftarrow 0$; $\tau \leftarrow \perp$; $c_0 \leftarrow \perp$; $s \leftarrow \perp$; $q \leftarrow$ "ref"
    $(sk_t^{[d\rangle}, pk_t^{[d\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % generate initial data key pair
    $(sk_t^{[\rho\rangle}, pk_t^{[\rho\rangle}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % initial refresh key pair
PROBE:
    $(q, X_{t+1}, s) \leftarrow \mathcal{A}_{KIChain}^{Oenc_{pk_\tau^{[d\rangle}}(\cdot), Odec_{b,t,\tau}(\cdot,\cdot)}(pk_t^{[d\rangle}, pk_t^{[\rho\rangle}, c_t, s)$
    while $q$ in $(\text{ref}, \text{test})$ do
        $(sk_{t+1}^{[d\rangle}, pk_{t+1}^{[d\rangle}) \leftarrow \mathcal{G}(1^k)$ % generate next data key pair
        $(sk_{t+1}^{[\rho\rangle}, pk_{t+1}^{[\rho\rangle}) \leftarrow \mathcal{G}(1^k)$ % generate next refresh key pair
        $c_{t+1} \leftarrow \mathcal{S}.\mathcal{E}_{pk_t^{[\rho\rangle}}(sk_{t+1}^{[d\rangle}, sk_{t+1}^{[\rho\rangle}, X_{t+1})$ % ciphertext of new secret keys plus suffix
        $t \leftarrow t + 1$
        if $b = 1$ and $q = \text{test}$ and $\tau = \perp$ then % only one key change period allowed
            $\tau \leftarrow t$
            $(\widehat{sk^{[d\rangle}}, \widehat{pk^{[d\rangle}}) \leftarrow \mathcal{S}.\mathcal{G}(1^k)$ % generate alternate data key pair
            $(q, X_{t+1}, s) \leftarrow \mathcal{A}_{KIChain}^{Oenc_{pk_\tau^{[d\rangle}}(\cdot), Odec_{b,t,\tau}(\cdot,\cdot)}(\widehat{pk^{[d\rangle}}, pk_t^{[\rho\rangle}, c_t, s)$
    else $(q, X_{t+1}, s) \leftarrow \mathcal{A}_{KIChain}^{Oenc_{pk_\tau^{[d\rangle}}(\cdot), Odec_{b,t,\tau}(\cdot,\cdot)}(pk_t^{[d\rangle}, pk_t^{[\rho\rangle}, c_t, s)$
DIST:
    $\hat{b} \leftarrow \mathcal{A}_{KIChain}^{Oenc_{pk_\tau^{[d\rangle}}(\cdot), Odec_{b,t,\tau}(\cdot,\cdot)}(s)$  % guess b
    return $\hat{b}$
```

We say that $\mathcal{A}_{KIChain}$ as above is a *KI chain* $\mathcal{S}$*-adversary* and define the *KI chain advantage* of $\mathcal{A}_{KIChain}$ against $\mathcal{S}$ as:

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}_{KIChain}}^{chain\text{–}weakM\text{–}ik\text{–}cca}(k) = Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{KIChain}}^{chain\text{–}weakM\text{–}ik\text{–}cca\text{–}1}(k) : \hat{b} = 1]$$
$$-Pr[(\tau, \hat{b}) \leftarrow \mathbf{Exp}_{\mathcal{S},\mathcal{A}_{KIChain}}^{chain\text{–}weakM\text{–}ik\text{–}cca\text{–}0}(k) : \hat{b} = 1] \,.$$

Claim. Any scheme $\mathcal{S}$ which is CCA semantically-secure and weak $IK - CCA$ secure for multiple challenges is also weak $IK - CCA$ chain secure.

Proof. This claim is similar to Lemma 8 and the proof has a similar structure to the proof in Section C.5. Generate all keys and key refreshes as in the experiment. Instead of giving $pk_t^{[d\rangle}$ to $\mathcal{A}_{KIChain}$ give key from key indistinguishability. experiment. If $\mathcal{A}_{KIChain}$ detects inconsistency between key and c_τ, then use it to build a GM adversary against $\mathcal{S}$. Otherwise, use oracles to answer $\mathcal{A}_{KIChain}$'s period τ encryption and decryption oracle queries. All other decryption oracle queries are handled with generated keys. Answer $\mathcal{A}_{KIChain}$'s guess as own in key indistinguishability experiment. Loss of advantage is one over number of chain periods.

Now using the claims above we complete the proof of Theorem 2. Indeed, since the public-key encryption scheme $\mathcal{S}$ used to implement $g\mathcal{IRC}$ is assumed to be both CCA semantically-secure and $IK - CCA$ (key indistinguishable) secure, the claims above collectively guarantee that $\mathcal{S}$ is also weak $IK - CCA$ chain secure. Assume then that there exists a successful splice adversary $\mathcal{A}$ against $g\mathcal{IRC}$. Using such an $\mathcal{A}$, we describe below how to build a successful weak $IK - CCA$ chain adversary $\mathcal{A}_{KIChain}$ against $\mathcal{S}$, which is a contradiction.

This reduction is quite similar to the one for proving Lemma 7 found in Appendix C.4, in which an intrusion-resilient channel adversary is converted into a chain adversary. Here $\mathcal{A}_{KIChain}$ simulates a channel for the splice adversary $\mathcal{A}$. As does the chain adversary in the proof of Lemma 7, $\mathcal{A}_{KIChain}$ guesses the challenge receiver u', the challenge period τ', and the graft point σ' for inserting the chain instance it is attacking into the simulated channel. $\mathcal{A}_{KIChain}$ likewise sets its chain challenge period $\lambda = \tau' - \sigma'$. The only significant difference here is that $\mathcal{A}_{KIChain}$ answers any encryption queries made by $\mathcal{A}$ of the form $(\cdot, \tau', \bar{u}')$ by querying its own encryption oracle.

E Two-Party Protocol Security: Proof of Lemma 3

Proof. Let $T = T(k)$ be the maximum number of time periods within a single execution of the channel $\mathcal{IRC}$ for security parameter k. For $t = 0, \ldots T$, let $\{Dist\}_t$ denote the set of all distinguishers which do not query for any forbidden messages for periods $< t$. By the assumption that Sim is a $\mathcal{Q}$-restricted, ε-good $\mathcal{Q}' \setminus \mathcal{Q}'_{FM}$-simulator for $\mathcal{P}$ it follows that Sim' is a $\mathcal{Q}$-restricted, ε-good $\mathcal{Q}' \setminus \mathcal{Q}'_{FM}$-simulator for $\mathcal{P}'$. Hence, $\forall D_T \in \{Dist\}_T$, the distinguishing advantage of D_T is negligible. Suppose for the sake of contradiction that there exists a distinguisher $D_0 \in \{Dist\}_0$ for Sim' and

$\mathcal{P}'$ with noticeable advantage δ. Then there must exist some $0 \leq t \leq T-1$ such that $\exists D_t \in \{Dist\}_t$ with noticeable advantage $\delta_1 \leq \delta$, but all $D_{t+1} \in \{Dist\}_{t+1}$ have negligible advantage.

We now describe how to use D_t to construct a $D_{t+1} \in \{Dist\}_{t+1}$ with noticeable advantage which is a contradiction. Since the restrictions on D_{t+1} and D_t are identical except for period t, D_{t+1} answers all non period t queries of D_t by using its own oracles. To answer D_t queries for period t, D_{t+1} does the following.

If D_t queries to expose some message sent by the refresh receiver, D_{t+1} simply queries its own oracle for this message.

If at some point D_t queries to expose some message sent by the refresh sender:

1. if D_{t+1} has previously exposed the refresh receiver, then it queries its own oracle for this exposure, since is not a forbidden message.
2. if D_{t+1} has previously exposed the refresh sender but not the refresh receiver, then it answers the query by selecting a message uniformly at random and querying its own channel oracle to obtain an encryption of this random message. Although this alters the view of D_k, the alteration is undetectable; see note below.
3. if D_{t+1} has not yet exposed the keys of either party, for the first such message query D_{t+1} guesses whether the refresh receiver will or will not in the end remain unexposed, that is whether such messages will or will not be forbidden. If D_{t+1} guesses the refresh receiver will later be exposed, it behaves as in case 1 above, for this and all subsequent such message queries. Otherwise, if D_{t+1} guesses the refresh receiver will remain unexposed, it encrypts randomness just as in case 2 above, for this and all subsequent such message queries.

If D_t queries to expose the refresh sender key, D_{t+1} likewise queries its own oracle to expose that party and passes the returned key on to D_t.

If D_t queries to expose the refresh receiver key, and has not previously exposed any messages sent to the refresh receiver, or has guessed (in case 3 above) that this key would become exposed, then D_{t+1} again queries its own oracle to expose this key. Otherwise, D_{t+1} aborts, since revealing the key would reveal all junk ciphertexts created by encrypting randomness.

Whenever D_{t+1} aborts it outputs a random bit $\hat{b} \xleftarrow{R} \{0,1\}$. If D_{t+1} does not abort its simulation for D_t, it waits and outputs whatever bit D_t outputs.

Conditioned on the event that D_{t+1} does not abort and provided that D_t can not detect the alteration in its view resulting from case 2 above, the distinguishing advantage of D_{t+1} will be within negligible of that of D_t. Since in a given execution D_{t+1} will not abort with probability at least 1/2, overall we have that the advantage of D_{t+1} is at least one-half of D_t's advantage, thus contradicting the original assumption on $\{Dist\}_{t+1}$.

It remains to be argued that no D_t can detect the alteration in its view resulting from case 2 above. Note that conditioned on the event that D_{t+1} does not abort its simulation, the view of D_t will be altered only in cases where D_t never exposes the refresh receiver for period t. Thus, it sufficient to show that no D_t can distinguish with noticeable advantage whether its view for period t contains encryptions of randomness under the revealed refresh sender key or encryptions of the requested messages under a random unrevealed key. The proof of this fact relies upon the assumption that the channel $\mathcal{IRC}$ is both spliceable and intrusion-resilient. Thus by Lemma 5 $\mathcal{IRC}$ is Key-Message-Splice-Indistinguishable. If D_k could distinguish between these two cases, then D_k could be used to build a Key-Message-Splice adversary D, which is a contradiction.

D works as follows. First, D generates an instance of protocol $\mathcal{P}$. Then, using its own channel oracles, D converts the $\mathcal{P}$ instance into a $\mathcal{P}'$ instance. Since by assumption D_k distinguishes without exposing the refresh receiver for period k, D will always be able to answer key exposure queries of D_k for periods $\leq k$ simply by querying its own channel oracles and adding any corresponding protocol $\mathcal{P}$ secrets. For refresh sender key exposure queries asked by D_k for periods $> k$, D gets the channel key from his own oracle. If the refresh receiver has not yet been exposed, D first applies $ChangeFMKey$ (see Appendix A.2) to this key. Then D hands the resulting alternate sender key to D_k together with the corresponding protocol $\mathcal{P}$ secret, and stores the alternate receiver key in order to consistently handle subsequent decryption queries of D_k, just as is done in the multi-splice version of Experiment $\mathbf{Exp}_{\mathcal{KEC},\mathcal{A}}^{ind\text{–}chan\text{–}splice\text{–}b}(k)$ (see Appendix A.2).

F Two-Party Protocol Security: Proof of Theorem 4 (SiBIR2)

Proof. Suppose to the contrary there exists a $\mathcal{Q}'$-restricted adversary $\mathcal{A}'$ with noticeable forgery success probability $\delta(k)$ against SiBIR2. Let $\mathcal{Q}$ be the class of all query sets $Q \in \mathcal{Q}'$ such that:

- The only exposure queries in Q are Signer key exposures and at most one Base exposure.
- There is at least one time period for which the Signer key is not Q-exposed[25].

Let $T = T(k)$ be the maximum number of time periods within a single execution of SiBIR1 or SiBIR2 for security parameter k. Plugging the simulator Sim for SiBIR1 from Lemma 10 (below) into Theorem 3 (Section 4.2) yields a $\mathcal{Q}$-restricted, $1/T$-good, $\mathcal{Q}'$ simulator Sim' for SiBIR2. Using Sim' and $\mathcal{A}'$ we build a $\mathcal{Q}$-restricted SiBIR1 adversary $\mathcal{A}$ as follows, contradicting Lemma 9 (below). $\mathcal{A}$ uses Sim' to simulate for $\mathcal{A}'$ and outputs the forgery attempt of $\mathcal{A}'$ as its own. Note as long as Sim' does not abort, $\mathcal{A}'$ maintains its δ success probability. Moreover, whenever Sim' picks the forgery period of $\mathcal{A}'$ as the period not to expose the Signer, Sim' does not abort, and if $\mathcal{A}'$ forges successfully so does $\mathcal{A}$. Thus with noticeable probability δ/T, $\mathcal{A}$ forges on SiBIR1.

Lemma 9. SiBIR1 *is intrusion-resilient against all $\mathcal{Q}$-restricted adversaries $\mathcal{A}$.*

Proof. Suppose there exists a $\mathcal{Q}$-restricted SiBIR1 adversary $\mathcal{A}$ with not negligible advantage. Let $\mathcal{A}_{SRSA}$ be the strong-RSA adversary which gets input of a remainder and modulus (α, n) as in the original SiBIR1 proof from [IR02]. Note that $\mathcal{A}_{SRSA}$ can simulate for any Q-restricted *asynchronous* $\mathcal{A}$ in an even simpler manner than the simulation described in [IR02] for any partially-synchronous adversary. This is because $\mathcal{A}$ here is $\mathcal{Q}$-restricted, and so is not allowed to query for any message exposures. In particular this assumption on $\mathcal{A}$ removes the requirement from [IR02] that the adversary be partially-synchronous in its query sequence. This assumption was needed on the adversary in [IR02] only due to the challenge of consistently answering message exposure queries for time periods prior to the forgery period. However, since $\mathcal{A}$ here is Q-restricted, there is no need to simulate any messages. The rest of the proof follows just as in [IR02]: the advantage of $\mathcal{A}_{SRSA}$ is related to the advantage of $\mathcal{A}$ precisely as shown in [IR02], thus contradicting the strong-RSA assumption.

Lemma 10 (SiBIR1 is simulatable). *There exists a $\mathcal{Q}$-restricted, $1/T$-good $\mathcal{Q}' \setminus \mathcal{Q}'_{FM}$ simulator for* SiBIR1*.*

Proof. Simulator Sim is defined as follows. Recall Sim gets the same interface with an instance of SiBIR1 as would a $\mathcal{Q}$-restricted forger.

1. Guess $j \in [1, T]$, the period for which the $\mathcal{Q}'$-restricted adversary will not expose the Signer.

[25] Of course this time period must precede any simultaneous exposure.

2. Exposes all Signer secrets up to and including $(j-1).(RN(j-1)+1)$.
3. Computes refresh and update messages for periods 1.1 through $(j-1).(RN(j-1))$ using the Signer secrets and public key.
4. Computes $b_{[1,T].0} = 1/s_{[1,T].0}$ and applies Base update (discarding update messages generated) and refresh algorithms using refresh messages previously computed from Signer long term secrets to compute Base long term secrets up to and including $b_{[j.(RN(j)+1)]}$.
5. Simultaneously exposes both Signer and Base secrets for period $(j+1).1$.
6. Runs SiBIR1 faithfully from this point (period $(j+1).1$) on to generate all subsequent secrets and messages.
7. forwards all $Osub$ (hash) and $Ofunc$ (signature) queries to his own oracles and returns the answers unchanged
8. Aborts if Signer becomes exposed for period j, or if a forbidden refresh/update is queried (Sim guessed incorrectly).

Note that Sim is $\mathcal{Q}$ restricted, exposing only Signer keys and one Base key (period j). By the same bijection argument found in [IR02] (unpublished appendix), if Sim does not abort, the distribution of views produced by Sim is identical to the distribution of views obtained from an honest instance of SiBIR1. Since the $\mathcal{Q}'$-restricted adversary must not Q-expose the signer key for at least one period, with probability $1/T$ period j is such a period and Sim simulates successfully.