*The proceedings version of this paper appears in ACM CCS 2007* [7]. *This is the full version of that paper.*

# Robust Computational Secret Sharing and a Unified Account of Classical Secret-Sharing Goals

Mihir Bellare*  Phillip Rogaway†

August 14, 2007

**Abstract**

We give a unified account of classical secret-sharing goals from a modern cryptographic vantage. Our treatment encompasses perfect, statistical, and computational secret sharing; static and dynamic adversaries; schemes with or without robustness; schemes where a participant recovers the secret and those where an external party does so. We then show that Krawczyk's 1993 protocol for robust computational secret sharing (RCSS) need not be secure, even in the random-oracle model and for threshold schemes, if the encryption primitive it uses satisfies only one-query indistinguishability (ind1), the only notion Krawczyk defines. Nonetheless, we show that the protocol *is* secure (in the random-oracle model, for threshold schemes) if the encryption scheme also satisfies one-query key-unrecoverability (key1). Since practical encryption schemes are ind1+key1 secure, our result effectively shows that Krawczyk's RCSS protocol is sound (in the random-oracle model, for threshold schemes). Finally, we prove the security for a variant of Krawczyk's protocol, in the standard model and for arbitrary access structures, assuming ind1 encryption and a statistically-hiding, weakly-binding commitment scheme.

**Key words:** Computational secret sharing, cryptographic protocols, provable security, robust computational secret sharing, secret sharing, survivable storage.

* Department of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093 USA. E-mail: mihir@cs.ucsd.edu WWW: www.cse.ucsd.edu/users/mihir/

† Department of Computer Science, University of California at Davis, Davis, California, 95616, USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand. E-mail: rogaway@cs.ucdavis.edu WWW: www.cs.ucdavis.edu/∼rogaway/

# Contents

# 1 Introduction

Work on classical secret-sharing tends to follow the traditions and sensibilities of information theory, combinatorics, or coding theory, not those of modern provable-security cryptography. Consider, for example, that the word *adversary* does not appear in the most widely cited survey of secret sharing [48]—but the word *information* appears some 50 times. Or consider that it was nearly 15 years *after* the invention of secret sharing by Blakley and Shamir [9, 43] until somebody, Krawczyk [31], made more than passing mention of the fact that there is a natural and useful complexity-theoretic setting for this problem. Even then, most subsequent work has ignored this "computational" setting.

In this paper we will recast classical secret-sharing in the tradition of provable-security cryptography. We will then use the freshened foundations to carry out a provable-security analysis of a well-known, useful, and formerly unanalyzed secret-sharing scheme. Before describing these contributions, we give some needed background.

BACKGROUND. In a robust computational secret sharing (RCSS) protocol, a dealer, assumed to be honest, breaks a secret $X$ into shares $X_1, \ldots, X_n$ and distributes them to $n$ players in such a way that an unauthorized set of players learns nothing about $X$ from their shares, yet an authorized set of players will reconstruct $X$, despite some players providing bogus shares, if and only if $X$ was shared. Both guarantees are computational, not information-theoretic. So RCSS relaxes the perfect secret-sharing goal of Shamir [43] in one dimension (computational privacy instead of information-theoretic privacy) and strengthens it in another (reconstructability in the face of incorrect shares, not just missing ones).

The RCSS goal, as well as a candidate solution, was invented by Krawczyk [31]. But Krawczyk provides no proofs or formal definitions for RCSS. Indeed his focus was not RCSS but CSS, computational secret-sharing, where recovery is for correct-or-missing shares. The CSS goal had been earlier mentioned by Karnin, Greene, and Hellman [30], who also consider the variant where cheating must be detected, not corrected. Robustness (recoverability despite some wrong shares) had already been studied in the information-theoretic setting by McEliece and Sarwate [35] and by Tompa and Woll [50].

Krawczyk's reason to look at CSS and RCSS was to reduce the size of participant shares: his mechanisms illustrate that, for threshold schemes, shares can be shorter than the secret, which is impossible in the information-theoretic setting [15, 30]. Krawczyk provides a CSS scheme with short shares using Rabin's idea of an *information-dispersal algorithm* (IDA) [40]. Robustness is then added-on using a hash-function-based technique that Krawczyk introduced in a separate paper [32]. Follow-on work to Krawczyk's paper has mostly focused on doing CSS for more general access structures [1, 14, 34, 51].

Protocols for CSS and RCSS are powerful tools or building secure and reliable distributed information-storage systems. A user's data (perhaps a file) is broken into pieces (shares) and stored on multiple servers in such a way that protects the privacy of the user from nosy servers, yet permits recovery of the data even if some of the servers provide invalid shares (either accidentally or intentionally). In recent years, and apparently without much notice from cryptographers, such systems and architectures have emerged from places like CMU and IBM [21, 28, 33, 39, 52]. Commercial product offerings and an open-source development community have also taken root.[1] An issue of *Computer* magazine explained these ideas [54]. Yet all of this has happened in the absence of even a formal *definition* for RCSS. In short, storage systems based on RCSS protocols already exist, but embody practice getting out in front of theory. As such, one cannot answer basic questions about these systems and their protocols, questions like "what exactly does this protocol *do*?" or "does CBC/IV=0 encryption suffice within it?"

OUR CONTRIBUTIONS. Coming at secret-sharing from a modern, provable-security angle, we make two contributions. One contribution is to revisit the basics of RCSS. We investigate the security of Krawczyk's RCSS protocol, which we call HK1. While Krawczyk made no formal definitions or claims in this regard, the only

[1] Examples include Cleversafe Corporation and the Cleversafe open-source user community (see http://www.cleversafe.org and http://www.cleversafe.com) and Security First Corporation (see http://securityfirstcorp.com).

| protocol | assume | and | access structure | result |
|---|---|---|---|---|
| HK1 | ind1 | random-oracle model | threshold | insecure (Sec. 4.2) |
| HK1 | ind1 + key1 | random-oracle model | threshold | secure (Th. 1, Th. 3) |
| HK2 | ind1 | statistically-hiding, weakly-binding commitment | arbitrary | secure (Th. 4, Th. 5) |

Figure 1: Summary of our results on Krawczyk's RCSS protocol (HK1) and a variant of it (HK2). By ind1 and key1 we mean one-query left-or-right indistinguishability and one-query key-unrecoverability.

---

encryption-scheme security property mentioned in his paper is the indistinguishability of $\mathit{Encrypt}_K(X)$ and $\mathit{Encrypt}_K(X')$, which we call *one-query indistinguishability* (ind1). Intuitively, this is all that HK1 should need, since, in the protocol, a key is used to encrypt just one message. Still, we show that HK1 is *not* secure under the assumption that its encryption scheme is ind1-secure, even for threshold schemes[2] and the random-oracle (RO) model [6]. Despite this, we show that HK1 *is* secure, for threshold schemes and in the RO model, if one assumes that the encryption scheme is ind1-secure *and* key1-secure, the latter being *one-query key-unrecoverability*. We complement this by proving ind1 + key1 to be the minimal assumption under which HK1 can be proved secure; see Appendix G. The assumption follows from *two-query indistinguishability* (ind2); see Proposition 6.Conventional encryption schemes are ind1- and key1-secure [3], so one may interpret our results as saying that, in the end, HK1 is sound, at least in the case of threshold schemes. The proof of security for HK2 is complex; intuitively, the complexity arises because one must sidestep the issues that cause an ind1-based instantiation of HK1 to fail. We go on to show that making a small change to HK1—replacing its hash-function by a noninteractive *statistically-hiding, weakly-binding* (SHWB) commitment-scheme—fixes all identified issues: the modified protocol, HK2, becomes provably secure for an arbitrary access structure, in the standard model, assuming just ind1-secure encryption. Our results are summarized in Figure 1.

To make the above results possible, we need a definition for RCSS. Not wanting to formalize yet another one-off secret-sharing notion, we show how to cast a large set of secret-sharing goals into a common framework. We give concrete-security, adversary-at-the-center definitions that encompass the perfect secret-sharing (PSS) goal of Shamir [43]; the less-than-perfect-privacy variant by Blakley [9]; the strengthening of PSS to robust schemes as envisioned by McEliece and Sarwate [35]; the alternative version of robustness described by Tompa and Woll [50]; and the relaxation of all this to the computational setting, as considered by Krawczyk [31]. Our definitions handle dynamic adversaries, apparently for the first time, and unify the information-theoretic and complexity-theoretic views. Look ahead to Figure 4 for a preview of some of the secret-sharing notions we encompass.

MORE ON DEFINITIONS. See Appendix B for a summary of existing PSS and CSS definitions [9, 31, 35, 43, 50], with and without robustness. The definitions frequently assume an *a priori* distribution on secrets, assume it to be the uniform over a large set, elide the syntax of a secret-sharing scheme, omit mention of any adversary, and make the implicit adversary static, with no simple way to make it dynamic.[3] The classical PSS definitions are so tailored to the perfect, information-theoretic case that there is no simple way to relax things to make a complexity-theoretic analog. Each definition is separate from each other, cut from its own cloth. No formal definition of the RCSS goal has ever appeared.

We aim to give a unified account of classical secret-sharing. To do this we define the privacy-advantage of an adversary $A$ attacking secret-sharing scheme $\Pi$, denoted $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$, and we define the recoverability-advantage of an adversary $B$ attacking a secret-sharing scheme $\Pi$, denoted $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(B)$, and we use these to define all notions of interest. For example, a secret-sharing scheme $\Pi$ is a PSS scheme if $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A) =$

---

[2] An $m$-out-of-$n$ threshold scheme is a secret-sharing scheme for which any $m$ uncorrupted players can recover the secret but smaller sets of players cannot. The set of sets of players authorized to recover the secret is the *access structure* for the scheme.

[3] A *static* adversary controls a certain set of players from the beginning, while a *dynamic* adversary chooses whom to corrupt as it corrupts players and learns their shares.

$\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(B) = 0$ for all "permissible" $A$ and $B$. There turn out to be four natural constraints on $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ and nine natural constraints on $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(B)$. Each classical secret-sharing notion shows up as one of the 36 combinations.

Our approach injects some order into the current definitional jungle of secret-sharing variants. In the process, we clarify that there have coexisted in the literature two fundamentally different notions of robustness. In the first, an uncorrupted player recovers the secret [50]; in the second, an external party has that job [35]. What is achievable in the two settings is vastly different (eg., external-party reconstructability can accommodate fewer corrupted players). It would seem that the two forms of robustness have coexisted in the literature for some 20 years without it even having being commented on that there *are* two kinds of robustness. Such a gap is probably attributable to the prior absence of a unifying viewpoint.

We comment that while our definitional framework is broad, it does not encompass verifiable secret-sharing (VSS) [17]. In a VSS scheme the dealer may be dishonest; for the goals in scope in this paper, the dealer is honest. Nor do we encompass proactive secret sharing [25], which, like VSS, has always been treated in the provable-security tradition. Our framework fails to encompass cheater detection or identification [12, 35], where the adversary is capable of obstructing recovery but incapable of forcing the recovery of a bogus secret. In this last case, however, our framework could certainly be extended to include these notions.

AFTERWARDS. After seeing a version of our paper, Yuval Ishai suggested a new RCSS protocol that combines a CSS protocol and a digital signature scheme [26]. Our intent in this paper was not to develop or analyze any fundamentally new protocol, but to analyze an existing protocol, HK1, that is already implemented, influential, and well-known. We also look at HK2 since it is a simple extension to HK1 that helps to shed light on it.

## 2 Preliminaries

ALGORITHMS AND ADVERSARIES. When we speak of an *algorithm* we mean an always-halting deterministic or probabilistic algorithm, possibly with access to one or more named oracles. A probabilistic algorithm can uniformly choose a random number between 1 and $i$ for an arbitrary positive integer $i$ by executing a statement $a \xleftarrow{\$} [i]$. If $A$ is an algorithm then $x \xleftarrow{\$} A(\cdots)$ means to choose $x$ according to the distribution induced by algorithm $A$, run on the elided arguments. If $A$ is deterministic we write $x \leftarrow A(\cdots)$ instead. If $A$ is a finite set then $x \xleftarrow{\$} A$ means to sample uniformly from it. If $A$ is a probabilistic algorithm then $x \in A(\cdot)$ means that $x$ occurs as an output with nonzero probability. We denote by $X_1 \ \cdots \ X_n$ or $X_1 \cdots X_n$ a reasonable encoding of $(X_1, \ldots, X_n)$ from which the constituents are uniquely recoverable. If the lengths of each $X_i$ is known then concatenation serves this purpose.

GAMES. We employ code-based game-playing in our proofs, as explored in [4]. In brief, a game is an always-halting program, written in code or pseudocode, that runs with an adversary. It specifies procedures *Initialize*, *Finalize*, and additional procedures (like *Deal*, $Corrupt$, and so forth), which are called *oracles*. In the code of a game, sets are initialized to empty and Booleans to `false`. The output of a game is the output of its Finalize procedure, or the output of the adversary itself if no Finalize is specified. We write $\Pr[G^A]$ for the probability that Finalize of game $G$ outputs `true` after the interaction with $A$.

ENCRYPTION SCHEMES. Adapting the formalization of [3], a symmetric *encryption scheme* is a pair of algorithms $\Pi^{\mathit{Enc}} = (\mathit{Encrypt}, \mathit{Decrypt})$ where *Encrypt* is a possibly probabilistic algorithm from $\{0,1\}^k \times \{0,1\}^*$ to $\{0,1\}^* \cup \{\perp\}$ and *Decrypt* is a deterministic algorithm from $\{0,1\}^k \times \{0,1\}^*$ to $\{0,1\}^* \cup \{\perp\}$. We call $k$ the *key length*. We write $\mathit{Encrypt}_K(X)$ and $\mathit{Decrypt}_K(Y)$ for $\mathit{Encrypt}(K, X)$ and $\mathit{Decrypt}(K, Y)$. We assume that whether or not $\mathit{Encrypt}_K(X) \in \{0,1\}^*$ (for $K \in \{0,1\}^k$) depends only on $|X|$ and we call the set of all $X$ such that $\mathit{Encrypt}_K(X) \in \{0,1\}^*$ the *domain* of $\Pi$. We require that if $Y \xleftarrow{\$} \mathit{Encrypt}_K(X)$ and $Y \neq \perp$ then $\mathit{Decrypt}_K(Y) = X$.

We define two notions of security for an encryption scheme $\Pi = (\mathit{Encrypt}, \mathit{Decrypt})$: indistinguishability (formalized in the left-or-right manner) and key-recoverability. For consistent syntax with the rest of this paper,

| | | |
|---|---|---|
| PROCEDURE *Initialize*<br>$K \xleftarrow{\$} \{0,1\}^k$<br>$b \xleftarrow{\$} \{0,1\}$<br><br>PROCEDURE *Finalize* $(d)$<br>RETURN $b = d$ | PROCEDURE *LeftOrRight*$(X^0, X^1)$<br>IF $\lvert X_0\rvert \neq \lvert X_1\rvert$ THEN RETURN $\perp$<br>$C \xleftarrow{\$} \mathit{Encrypt}_K(X^b)$<br>RETURN $C$ | Game Ind |
| PROCEDURE *Initialize*<br>$K \xleftarrow{\$} \{0,1\}^k$<br><br>PROCEDURE *Finalize* $(K')$<br>RETURN $K = K'$ | PROCEDURE *Enc*$(X)$<br>$C \xleftarrow{\$} \mathit{Encrypt}_K(X)$<br>RETURN $C$ | Game Key |

Figure 2: Games used to define the privacy of an encryption scheme $\Pi = (\mathit{Encrypt}, \mathit{Decrypt})$.

we describe both notions using games. See Figure 2 for the definitions of these games, named Ind and Key. Based on them, define the indistinguishability advantage by $\mathbf{Adv}_{\Pi}^{\mathrm{ind}}(A) = 2\Pr[\mathrm{Ind}^A] - 1$. The notion is the same as in [3]. We let $\mathbf{Adv}_{\Pi}^{\mathrm{key}}(A) = \Pr[\mathrm{Key}^A]$ be the probability that $A$ recovers the encryption key.

# 3 The Definitional Framework

In this section we unify and extend definitions in the literature for perfect secret-sharing and computational secret-sharing, both with and without robustness. We break with tradition by handling information-theoretic secret-sharing neither in terms of entropy nor equality of distributions, but in a way that directly models and measures the adversary's aims. Also breaking with tradition, we directly handle dynamic adversaries. For ease of comparison, some traditional secret-sharing definitions are recalled in Appendix B. We warn that, to achieve our desired level of generality, this section is more dense and atypical than it would be if were we *just* trying to define Krawczyk-style RCSS (entry CSS-CR2 in Figure 4).

OVERVIEW. Secret-sharing schemes have two basic requirements: *privacy* and *recoverability* (the latter is also called *reconstructability*). Privacy entails that an unauthorized coalition of players can't learn anything about the secret that's been shared. It can be *complexity-theoretic* or *information-theoretic*. Information-theoretic schemes maintain privacy no matter how much computing power the adversary has; complexity-theoretic ones protect the privacy of the shared secret from adversaries with "reasonable" computing resources. In the information-theoretic setting, security can be *perfect* (absolutely no information is revealed about the secret) or possibly less than perfect, which is called *statistical* privacy. The adversary that is attacking a scheme's privacy can be *static* (it decides which players to corrupt at the beginning of its attack) or *dynamic* (it chooses which players to attack one-by-one, as it learns shares). Our definition of the *privacy advantage* that an adversary $A$ gets in attacking a secret-sharing scheme $\Pi$, denoted $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$, encompass and measures all of the above possibilities.

Recoverability entails that authorized coalitions of players can reconstruct the secret. It can be guaranteed in the *erasure model* or the *substitution model*. In the erasure model, the adversary marks shares of corrupted players as *missing* but cannot otherwise modify a player's share.[4] Secret-sharing schemes secure in the substitution model, where the adversary *may* modify a corrupted player's share, are called *robust*. Preserving a distinction with us since [35, 50], we distinguish two flavors of robustness: the shared secret can be recovered by an *uncorrupted player* or by an *external party*. It is easier for an uncorrupted player to recover the secret than for an external party to do so since an uncorrupted player knows one particular share—his own—that he can

[4] One could distinguish two variants: the adversary *must* mark the shares of corrupted players as missing, or the adversary *may* mark the shares of corrupted players as missing (or may leave them unchanged). We assume the former.

| | |
|---|---|
| PROCEDURE *Deal*$(S^0,\, S^1)$ | PROCEDURE $Corrupt(i)$ Game Priv |
| IF NOT $\boldsymbol{S}$ THEN $b \xleftarrow{\$} \{0,1\}$, $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}(S^b)$ | $T \leftarrow T \cup \{i\}$ |
| RETURN | RETURN $\boldsymbol{S}[i]$ |
| PROCEDURE *Finalize* $(d)$ | |
| RETURN $b = d$ | |

| | |
|---|---|
| PROCEDURE *Deal*$(S)$ | PROCEDURE $Corrupt(i)$ Game Rec |
| IF NOT $\boldsymbol{S}$ THEN $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}(S)$ | $T \leftarrow T \cup \{i\}$ |
| RETURN | RETURN $\boldsymbol{S}[i]$ |
| PROCEDURE *Finalize* $(\boldsymbol{S}', j)$ | |
| RETURN $\mathit{Recover}(\boldsymbol{S}_{\overline{T}} \sqcup \boldsymbol{S}'_T, j) \neq S$ | |

Figure 3: Games used to define privacy and recoverability of secret-sharing scheme $\Pi = (\mathit{Share}, \mathit{Recover})$.

---

assume to be right (remember that the types of secret sharing dealt with in this paper assume an honest dealer). As before, a recoverability-attacking adversary may be static or dynamic. Our definition of the *recoverability advantage* that an adversary $A$ gets in attacking a secret-sharing scheme $\Pi$, denoted $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A)$, encompass and measures all of the above possibilities. To accomplish this, we regard the erasure model as a special class of adversaries, Rec$\Diamond$, where any $A \in$ Rec$\Diamond$ replaces the shares of corrupted players with the distinguished value $\Diamond$ (missing). We likewise regard recovery-by-an-uncorrupted player as a special class of adversaries, Rec1, where an $A \in$ Rec1 is obliged to output the identity of some uncorrupted player $j$. Adversaries that may arbitrarily substitute shares for corrupted players live live in the class Rec.

We will define notions in a way that permits consideration of an arbitrary access structure. Indeed we will be more general still, defining privacy and recoverability in a way that depends on an arbitrary set of adversaries.

To simplify and strengthen definitions and theorem statements, we focus on concrete (as opposed to asymptotic) definitions. But we do explain how to lift the definitions to the asymptotic setting.

SYNTAX. An $n$-party *secret-sharing scheme* with message space $\mathbb{S}$ is a pair $\Pi = (\mathit{Share}, \mathit{Recover})$. Here *Share* is a probabilistic algorithm that, on input $S \in \mathbb{S}$ returns the $n$-vector $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}(S)$ where each $\boldsymbol{S}[i] \in \{0,1\}^*$ and *Recover* is a deterministic algorithm that on input $\boldsymbol{S} \in \left(\{0,1\}^* \cup \{\Diamond\}\right)^n$ and $j \in [0\,..\,n]$ returns a value $S \leftarrow \mathit{Recover}(\boldsymbol{S}, j)$ where $S \in \mathbb{S} \cup \{\Diamond\}$. We assume $\mathit{Share}(S)$ returns $\perp$ ("undefined") if $S \notin \mathbb{S}$.

Let us explain the intent of the syntax. A secret-sharing scheme specifies two different algorithms. The first, *Share*, is used by a *dealer* who wants to distribute some secret $S \in \mathbb{S}$ to a group of $n$ players, numbered $1, \ldots, n$. The dealer applies *Share* to the secret $S$. The result is a vector $\boldsymbol{S} = (\boldsymbol{S}[1], \ldots, \boldsymbol{S}[n])$ with each share $\boldsymbol{S}[i]$ a string. The dealer gives $\boldsymbol{S}[i]$ to party $i$. As *Share* is probabilistic, different runs of $\mathit{Share}(S)$ may return different vectors. When, at some later point, an entity would like to recover the secret, it must first try to collect up enough shares. It forms an $n$-element vector $\boldsymbol{S} = (\boldsymbol{S}[1], \ldots, \boldsymbol{S}[n])$. The $i^{\mathrm{th}}$ component of this vector, $\boldsymbol{S}[i]$, is either a string $\boldsymbol{S}[i] \in \{0,1\}^*$ or the distinguished value $\Diamond$. In the first case the value $\boldsymbol{S}[i]$ is the *purported* share of party $i$ while in the second case the share $\boldsymbol{S}[i] = \Diamond$ has been marked as *missing*. The party who wants to recover the shared secret now applies the algorithm *Recover* to the vector $\boldsymbol{S}$ and a number $j \in [0\,..\,n]$, the number indicating the location of a share that is *known* to be valid. If no particular share is known valid, set $j = 0$ and write $\mathit{Recover}(\boldsymbol{S})$ for $\mathit{Recover}(\boldsymbol{S}, 0)$. To make sense, one must have $\boldsymbol{S}[j] \neq \Diamond$ if $j \in [n] = [1\,..\,n]$. The value that emerges from applying *Recover* will be either the recovered secret $S \in \mathbb{S}$ or the distinguished value $\Diamond$. The latter indicates that the algorithm is unable to recover the underlying secret.

PRIVACY. Let $\Pi = (\mathit{Share}, \mathit{Recover})$ be an $n$-party secret-sharing scheme with message space $\mathbb{S}$. Let $A$ be an adversary. We consider the privacy game Priv of Figure 3. To run $A$ with Priv the following happens. First, initialize $T \leftarrow \emptyset$. Now run $A$. It should first make an oracle call $\mathit{Deal}(S^0, S^1)$ satisfying $S^0, S^1 \in \mathbb{S}$ and $|S^0| = |S^1|$. The game then chooses a hidden bit $b$ and samples $\boldsymbol{S}$ from $\mathit{Share}(S^b)$. Nothing is returned to $A$ in response to its query. Next the adversary $A$ makes oracle queries of the form $Corrupt(i)$ where

| fullname (nick-name) | $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A)$ | when $A$ is in | $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A)$ | when $A$ is in | reference |
|---|---|---|---|---|---|
| PSS-PR0 (PSS) | 0 | $\mathcal{A} \cap \mathsf{Priv}$ | 0 | $\mathcal{A} \cap \mathsf{Rec}\Diamond$ | Shamir [43] |
| PSS-PR2 | 0 | $\mathcal{A} \cap \mathsf{Priv}$ | 0 | $\mathcal{A} \cap \mathsf{Rec}$ | McEliece & Sarwate [35] |
| PSS-SR1 | 0 | $\mathcal{A} \cap \mathsf{Priv}$ | *small* | $\mathcal{A} \cap \mathsf{Rec1}$ | Tompa & Woll [50] |
| PSS-SR2 | 0 | $\mathcal{A} \cap \mathsf{Priv}$ | *small* | $\mathcal{A} \cap \mathsf{Rec}$ | Rabin & Ben-Or [41] |
| SSS-PR0 | *small* | $\mathcal{A} \cap \mathsf{Priv}$ | 0 | $\mathcal{A} \cap \mathsf{Rec}\Diamond$ | Blakley [9] |
| CSS-PR0 (CSS) | *small* | $\mathcal{A} \cap \mathsf{Priv} \cap \mathsf{Prac}$ | 0 | $\mathcal{A} \cap \mathsf{Rec}\Diamond$ | Krawczyk [31] |
| CSS-CR1 | *small* | $\mathcal{A} \cap \mathsf{Priv} \cap \mathsf{Prac}$ | *small* | $\mathcal{A} \cap \mathsf{Rec1} \cap \mathsf{Prac}$ | apparently new |
| CSS-CR2 (RCSS) | *small* | $\mathcal{A} \cap \mathsf{Priv} \cap \mathsf{Prac}$ | *small* | $\mathcal{A} \cap \mathsf{Rec} \cap \mathsf{Prac}$ | Krawczyk[31] |
| NSS-PR0 (IDA) | — | — | 0 | $\mathcal{A} \cap \mathsf{Rec}\Diamond$ | Rabin [40] |
| NSS-PR1 | — | — | 0 | $\mathcal{A} \cap \mathsf{Rec1}$ | Witsenhausen [53] |
| NSS-PR2 (ECC) | — | — | 0 | $\mathcal{A} \cap \mathsf{Rec}$ | Shannon [44] |

Figure 4: Selected ways of combining $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A)$ and $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A)$ constraints to recover significant definitions. For some notions it is conventional to also demand that $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) = 0$ for all $A \in \mathcal{A} \cap \mathsf{Rec}\Diamond$.

---

$i \in [n]$. The query is a request to *corrupt* the indicated player. In response to query $Corrupt(i)$ the game sets $T \leftarrow T \cup \{i\}$ and returns share $\boldsymbol{S}[i]$. When $A$ is done corrupting players it outputs a bit $d$ and halts. It is said to *win* if $b = d$. We measure its success as twice the probability of its winning minus one; formally, $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) = 2\Pr[\mathrm{Priv}^A] - 1$. Let $\mathsf{Priv}$ be the class of adversaries, the *privacy adversaries*, that behave as we have just described, regardless of oracle responses.

RECOVERABILITY. Fix an $n$-party secret-sharing scheme $\Pi = (\mathit{Share}, \mathit{Recover})$ with message space $\mathbb{S}$. Let $A$ be an adversary. We consider the recoverability game Rec of Figure 3. First, initialize $T \leftarrow \emptyset$. Now run adversary $A$. It should first call $\mathit{Deal}(S)$ for some $S \in \mathbb{S}$. Note that *Deal* takes just one argument this time. The game then selects an $n$-vector $\boldsymbol{S}$ from $\mathit{Share}(S)$. Next the adversary corrupts players. Each time it calls $Corrupt(i)$ the game sets $T \leftarrow T \cup \{i\}$ and returns $\boldsymbol{S}[i]$. When the adversary is done corrupting players it outputs a pair $(\boldsymbol{S}', j)$ where $j \in [0\,..\,n] \setminus T$ and $\boldsymbol{S}' \in (\{0,1\}^* \cup \{\Diamond\})^n$. Let $\boldsymbol{S}_{\overline{T}} \sqcup \boldsymbol{S}'_T$ be the $n$-vector whose $i^{\mathrm{th}}$ component is $\boldsymbol{S}'[i]$ if $i \in T$ and $\boldsymbol{S}[i]$ otherwise. The adversary is said to *win* if $\mathit{Recover}(\boldsymbol{S}_{\overline{T}} \sqcup \boldsymbol{S}'_T, j) \neq S$. We measure the adversary's success by the real number $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) = \Pr[\mathrm{Rec}^A]$. Let $\mathsf{Rec}$ be the class of adversaries, the *recoverability adversaries*, that behave as we have just described, regardless of oracle responses.

We define a set $\mathsf{Rec}\Diamond \subseteq \mathsf{Rec}$, the *erasure adversaries*. Adversary $A \in \mathsf{Rec}$ is in $\mathsf{Rec}\Diamond$ if, whenever $A$ outputs $(\boldsymbol{S}', j)$, we have $\boldsymbol{S}'[i] = \Diamond$ for all $i \in [n]$: the adversary replaces the shares of corrupted players by $\Diamond$. Similarly, we define a set $\mathsf{Rec1} \subseteq \mathsf{Rec}$, the *recoverability-1 adversaries*. Adversary $A \in \mathsf{Rec}$ is in $\mathsf{Rec1}$ if, whenever $A$ outputs $(\boldsymbol{S}', j)$, we have $j > 0$ and $j$ is uncorrupted. The adversary is obliged to point to an uncorrupted player. As a mnemonic, the adversary must identify one good player.

SECRET-SHARING DEFINITIONS. Let $\Pi = (\mathit{Share}, \mathit{Recover})$ be secret-sharing scheme and let $\mathcal{A}$ be a class of adversaries. We can demand $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A)$ be: PSS: *zero* for any privacy adversaries in $\mathcal{A}$; SSS: *small* for *any* privacy adversary in $\mathcal{A}$; CSS: small for any *practical* privacy adversary in $\mathcal{A}$; or NSS: *no* privacy demands at all. (Letters P, S, C, and N stand for *perfect*, *statistical*, *computational*, and *none*, while SS is for *secret sharing*.) Similarly, we can demand $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A)$ be: PR0: zero for any *erasure* adversary in $\mathcal{A}$; PR1: zero for any *recoverability-1* adversary in $\mathcal{A}$; PR2: zero for *any* recoverability adversary in $\mathcal{A}$; SR0: small for for any erasure adversary in $\mathcal{A}$; SR1: small for any recoverability-1 adversary in $\mathcal{A}$; SR2: small for any recoverability adversary in $\mathcal{A}$; CR0: small for any practical erasure adversary in $\mathcal{A}$; CR1: small for any practical recoverability-1 adversary in $\mathcal{A}$; or CR2: small for any practical recoverability adversary in $\mathcal{A}$.

(Letters P, S, and C are as before, and R is for robustness.) All in all there are $4 \cdot 9 = 36$ notions obtained by combining the named requirements on $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ and $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(A)$. We single out some of them in Figure 4.

Several entries in the table are familiar, and some go by other names; these are credited, where appropriate, to the party associated to the basic notion. Some notions are not conventionally regarded as secret-sharing yet show up in the table: error-correcting codes and Rabin's information dispersal algorithms [40].

(As we will be using IDAs and ECCs, let us pause and give a concrete instantiation. The simplest IDA is based on replication: $\mathit{Share}(X) = (X, \ldots, X)$ and $\mathit{Recover}((X_1, \ldots, X_n), j) = X$ if $\{ \boldsymbol{X}[i] \ : \ \boldsymbol{X}[i] \neq \Diamond \} = \{X\}$ while $\mathit{Recover}((X_1, \ldots, X_n), j) = \Diamond$ otherwise. IDAs with shorter share lengths also exist [40]. A simple ECC scheme again uses replication: $\mathit{Share}(X) = (X, \ldots, X)$ and $\mathit{Recover}(X_1, \ldots, X_n) = X$ if there is a string $X$ that occurs more than $n/2$ times among $X_1, \ldots, X_n$, and $\mathit{Recover}(X_1, \ldots, X_n) = \Diamond$ otherwise. When $\mathcal{A} \cap \mathsf{Rec} \subseteq \mathsf{Rec1}$ we can change this to $\mathit{Share}(X) = (X, \ldots, X)$ and $\mathit{Recover}((X_1, \ldots, X_n), j) = X_j$ if $X_j \neq \Diamond$ and $\mathit{Recover}((X_1, \ldots, X_n), j) = \Diamond$ if $X_j = \Diamond$.)

Secret-sharing scheme$\Pi$ has *perfect privacy* over $\mathcal{A}$ if $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A) = 0$ for all $A \in \mathcal{A}$, and it has *perfect recoverability* over $\mathcal{A}$ if $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(A) = 0$ for all $A \in \mathcal{A}$. Figure 4 serves to rigorously define PSS-PR0 (PSS), PSS-PR2, NSS-PR0 (IDA), NSS-PR1, and NSS-PR2: for example, $\Pi$ is a PSS with respect to $\mathcal{A}$ if $\Pi$ has perfect privacy over $\mathcal{A} \cap \mathsf{Priv}$ and perfect recoverability over $\mathcal{A} \cap \mathsf{Rec}\Diamond$.

The remaining seven rows of Figure 4 contain *small* or $\mathsf{Prac}$, which we haven't yet described. For the statistical notions (*small* and no $\mathsf{Prac}$) one can introduce a real number in place of *small* [50]. For example, an $\epsilon$-robust PSS-SR1 scheme $\Pi$ over $\mathcal{A}$ has perfect privacy over $\mathcal{A}$ and $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(A) \leq \epsilon$ for all $A \in \mathcal{A} \cap \mathsf{Rec1}$.

For the computational goals there are two options. One is to leave the security notion formally undefined but make concrete-security statements to bound $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ or $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(A)$ in terms of other quantities. This is the concrete-security approach, and we adopt it for Theorems 1–5.

A different option (which applies to any of the 36 notions) is to move to the asymptotic setting. For this one adds in a security parameter $k$ and interprets *small* in Figure 4 as *negligible* (vanishing faster than the inverse of any polynomial) and interprets $\mathsf{Prac}$ as the class of probabilistic polynomial time (PPT) algorithms. A secret-sharing scheme now involves $n(k)$ parties and has a message space $\mathbb{S}(k) \subseteq \{0,1\}^*$. The *Share* and *Recover* algorithms are polynomial-time that take an additional (first) input of $1^k$. Adversary $A$ is likewise provided $1^k$. Advantage measures $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ and $\mathbf{Adv}_{\Pi}^{\mathrm{rec}}(A)$ of an adversary $A$ become functions of $k$. Note that in moving to the asymptotic setting we do not use the length of the secret as the security parameter, a questionable definitional choice in some prior treatments. See Appendix B.

ACCESS STRUCTURES. We defined secret-sharing goals with respect to an adversary class, but the classical approach is to use an access structure instead. Our approach is more general (and the added generality is needed to encompass contexts like that of McEliece and Sarwate [35]). An $n$-party *access structure* is a set $\mathcal{A}$ of subsets of $[n]$ that is *monotone*: if $R \subseteq S \subseteq [n]$ and $R \in \mathcal{A}$ then $S \in \mathcal{A}$. Each $S \in \mathcal{A}$ is said to be *authorized*. The most common access structure is the threshold access structure $\mathcal{A}_{m,n}$ where $m, n \geq 1$ and $0 \leq m \leq n$. This is the access structure defined by saying that $S \in \mathcal{A}_{m,n}$ iff $S \subseteq [n]$ and $|S| \geq m$.

We associate to any $n$-party access structure $\mathcal{A}$ two classes of adversaries. The first, $\mathcal{A}^{\mathrm{p}}$, is all privacy adversaries $A$ that never corrupt an authorized set ($A$ never corrupts a set $S \in \mathcal{A}$). The second, $\mathcal{A}^{\mathrm{r}}$, is all recoverability adversaries $A$ that always leave uncorrupted an authorized set (if $A$ corrupts $T$ then $[n]\backslash T \in \mathcal{A}$).[5] In speaking of the players that $A$ can corrupt, we quantify over all possible oracle responses (not necessarily those associated to any particular game). Corrupting $i$ means calling $Corrupt(i)$. The asymmetry embodied in the $\mathcal{A}^{\mathrm{p}}$ and $\mathcal{A}^{\mathrm{r}}$ definitions arises because privacy is unachievable if *some* authorized set of players gets *corrupted* while robustness is unachievable if *no* authorized set of players remains *uncorrupted*.

To access structure $\mathcal{A}$ we associate adversary class $\mathcal{A}^{\mathrm{p}} \cup \mathcal{A}^{\mathrm{r}}$, which we also call $\mathcal{A}$. In this way, any definition over an adversary class provides the corresponding definition over an access structure.

VALID ADVERSARIES. For our robustness results we need a technical condition on the class of adversaries that

[5] These notions are not the same. As an example, for threshold schemes, $\mathcal{A}_{m,n}^{\mathrm{p}}$ is the set of privacy adversaries that corrupt at most $m-1$ players, while $\mathcal{A}_{m,n}^{\mathrm{r}}$ is the set of recoverability adversaries that corrupt at most $n-m$ players.

can be handled. First, say that adversary $A \in \mathsf{Rec}$ can *generate* $(S, \boldsymbol{S}, T, \boldsymbol{S}', j)$ if it can call *Deal*$(S)$, resulting in shares $\boldsymbol{S}$, corrupt players $T \subseteq [n]$, and output $(\boldsymbol{S}', j)$. We say $(S, \boldsymbol{S}, T, \boldsymbol{S}', j)$ is $\mathcal{A}$-*generable* if there is an $A \in \mathcal{A} \cap \mathsf{Rec}$ such that $A$ can generate $(S, \boldsymbol{S}, T, \boldsymbol{S}', j)$. Now for $\boldsymbol{S}', \boldsymbol{S}'' \in (\{0,1\}^* \cup \{\Diamond\})^n$ let us say that $\boldsymbol{S}' \geq \boldsymbol{S}''$ ($\boldsymbol{S}'$ is worse than $\boldsymbol{S}''$) if $\boldsymbol{S}'[i] = \Diamond$ implies $\boldsymbol{S}''[i] = \Diamond$. We say that $\mathcal{A} \subseteq \mathsf{Rec}$ is *valid* (with respect to some secret-sharing scheme $\Pi$) if the following is true: if $(S, \boldsymbol{S}, T, \boldsymbol{S}', j)$ is $\mathcal{A}$-generable and $\boldsymbol{S}' \geq \boldsymbol{S}''$ then the following adversary $A_{S,T,\boldsymbol{S}',j,\boldsymbol{S}''}$ is in $\mathcal{A}$: it calls *Deal*$(S)$; then it calls $Corrupt(i)$ for each $i \in T$ (say in numerical order); then it outputs $(\boldsymbol{S}'', j)$. Intuitively, if an adversary is allowed to provide a bogus share $\boldsymbol{S}'[i]$ of $S \in \{0,1\}^*$ it should be allowed to provide a bogus share $\boldsymbol{S}''[i] \in \{0,1\}^* \cup \{\Diamond\}$ of $S$.

The class $\mathcal{A}^{\mathrm{r}}$ associated to any access structure $\mathcal{A}$ is valid. So too is $\mathcal{A}_{m,n,t} \cap \mathsf{Rec}$ where $\mathcal{A}_{m,n,t}$ [35] is $\mathcal{A}^{\mathrm{p}}_{m,n} \cup (\mathcal{A}^{\mathrm{r}}_{m,n} \cap \mathcal{A}_t)$ and $\mathcal{A}_t$ is adversaries that can only output $(\boldsymbol{S}', j)$ with $\boldsymbol{S}'$ having at most $t$ non-$\Diamond$ components. Thus $A \in \mathcal{A}_{m,n,t}$ is a privacy adversary that can corrupt at most $m-1$ players *or* a recoverability adversary that can corrupt at most $n-m$ players, replacing at most $t$ shares with strings and the rest with $\Diamond$.

EXTENSIONS. One can augment a secret-sharing scheme by allowing a *Setup* algorithm; we would now have a triple of algorithms $\Pi = (\mathit{Setup}, \mathit{Share}, \mathit{Recover})$. *Setup* is probabilistic and outputs a *public parameter* $P \in \{0,1\}^*$. Procedures *Share* and *Recover* are provided $P$, as is any adversary attacking the scheme. While *Share* could always install the public parameter in each player's share, the effect is not the same as adding a *Setup*: in one setting, the adversary has to corrupt a player to get $P$ and in the other it is free; and there are important efficiency-accounting consequences, as pulling out the public parameter might shorten the shares.

Our privacy and authenticity notions can be lifted to the random-oracle setting [6]. To do so, add to games Priv and Rec a procedure *Hash* that realizes a random function from strings of arbitrary length to strings of some desired length. Algorithms *Share* and *Recover* are allowed to call *Hash*, as may the adversary itself.

Our notions of privacy and recoverability consider an adversary that can obtain the deal of only one secret. One can easily extend our definitions to handle the sharing of multiple secrets. A standard hybrid argument can be used to show that the two definitions are equivalent (up to a multiplicative factor of the number of secrets dealt). This result depends on the *Share* algorithm being stateless, as it is for all the schemes of this paper. If *Share* is stateful, a natural counter-example shows that the deal-one-secret and deal-multiple-secret notions are inequivalent.

STATIC ADVERSARIES. Classical definitions of secret sharing assume a static adversary. This is encompassed by our framework in the sense that it is easy to restrict attention to static adversaries. Let $\mathsf{Static}$ be the set of all adversaries $A$ for which there is a set $T$ associated to $A$ such that, regardless of $A$'s input, coins, and oracle responses, the set of players corrupted by $A$ is $T$. To consider static adversaries restrict to sets like $\mathsf{Priv} \cap \mathsf{Static}$. A static adversary $A$ can be imagined to deterministically "decide" at the beginning of its execution which players $T$ to corrupt. We define adversaries

# 4 The HK1 Protocol (Krawczyk's RCSS Scheme)

## 4.1 Krawczyk's construction

We reproduce Krawczyk's construction using our notation. Fix a family of adversaries $\mathcal{A}$. We build an $n$-party secret-sharing scheme with message space $\mathbb{S}$ from the five components: (1) a symmetric encryption scheme $\Pi^{\mathit{Enc}} = (\mathit{Encrypt}, \mathit{Decrypt})$ with $k$-bit keys and message space $\mathbb{S}$; (2) an $n$-party PSS $\Pi^{\mathit{PSS}} = (\mathit{Share}^{\mathit{PSS}}, \mathit{Recover}^{\mathit{PSS}})$ over $\mathcal{A}$ with message space $\{0,1\}^k$; (3) an $n$-party IDA $\Pi^{\mathit{IDA}} = (\mathit{Share}^{\mathit{IDA}}, \mathit{Recover}^{\mathit{IDA}})$ over $\mathcal{A}$ with message space $\Sigma^*$; (4) an $n$-party ECC $\Pi^{\mathit{ECC}} = (\mathit{Share}^{\mathit{ECC}}, \mathit{Recover}^{\mathit{ECC}})$ over $\mathcal{A}$ with message space $\{0,1\}^h$; and (5) a function $\mathit{Hash}\colon \{0,1\}^* \to \{0,1\}^h$. We call $\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \mathit{Hash}$ the underlying *primitives* of the HK1 scheme, and say that they are over $\mathcal{A}$, for $n$ parties and for $h$-bit hashes. From such a set of primitives define $\mathrm{HK1}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \mathit{Hash}] = (\mathit{Share}, \mathit{Recover})$ as specified and illustrated in Figure 5. In its line 21, if $\boldsymbol{X}[i] = \Diamond$ then our convention is to assign $\Diamond$ to all variables on the left-hand side of the assignment statement; otherwise $\boldsymbol{X}[i]$ is parsed into its corresponding, uniquely defined constituents. Similarly, if $K = \Diamond$

PROCEDURE $\mathit{Share}(X)$
10 $K \xleftarrow{\$} \{0,1\}^k$ ; $C \xleftarrow{\$} \mathit{Encrypt}_K(X)$
11 $\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(K)$
12 $\boldsymbol{C} \xleftarrow{\$} \mathit{Share}^{\mathit{IDA}}(C)$
13 FOR $i \leftarrow 1$ TO $n$ DO
14 $\quad \boldsymbol{H}[i] \leftarrow \mathit{Hash}(\boldsymbol{K}[i]\,\boldsymbol{C}[i])$
15 $\quad \boldsymbol{S}_i \xleftarrow{\$} \mathit{Share}^{\mathit{ECC}}(\boldsymbol{H}[i])$
16 FOR $i \leftarrow 1$ TO $n$ DO
17 $\quad \boldsymbol{X}[i] \leftarrow \boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i]$
18 RETURN $\boldsymbol{X}$

PROCEDURE $\mathit{Recover}(\boldsymbol{X}, j)$
20 FOR $i \leftarrow 1$ TO $n$ DO
21 $\quad \boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i] \leftarrow \boldsymbol{X}[i]$
22 FOR $i \leftarrow 1$ TO $n$ DO
23 $\quad \boldsymbol{H}[i] \leftarrow \mathit{Recover}^{\mathit{ECC}}(\boldsymbol{S}_i, j)$
24 FOR $i \leftarrow 1$ TO $n$ DO
25 $\quad$ IF $\boldsymbol{X}[i] \neq \Diamond$ AND $\mathit{Hash}(\boldsymbol{K}[i]\,\boldsymbol{C}[i]) \neq \boldsymbol{H}[i]$
26 $\quad\quad$ THEN $\boldsymbol{K}[i] \leftarrow \Diamond$ ; $\boldsymbol{C}[i] \leftarrow \Diamond$
27 $K \leftarrow \mathit{Recover}^{\mathit{PSS}}(\boldsymbol{K}, j)$
28 $C \leftarrow \mathit{Recover}^{\mathit{IDA}}(\boldsymbol{C}, j)$
29 $X \leftarrow \mathit{Decrypt}_K(C)$
30 RETURN $X$

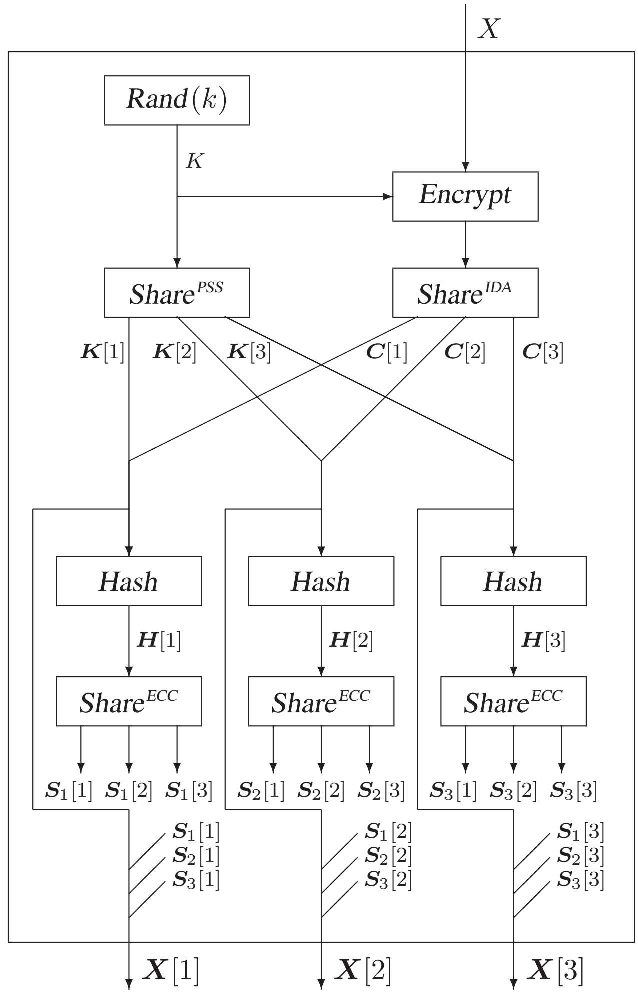


Figure 5: *Left*: Definition of the HK1 construction $\Pi = (\mathit{Share}, \mathit{Recover}) = \text{HK1}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \mathit{Hash}]$. *Right*: Illustration of the scheme's *Share* algorithm for $n = 3$ players. Procedure *Rand*, on input $k$, returns a uniformly random $k$-bit string.

---

or $C = \Diamond$ when line 29 is executed then our convention is that $X = \Diamond$. Let $\text{HK1}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}] = (\mathit{Share}, \mathit{Recover})$ be the random-oracle variant of this scheme in which $\mathit{Hash}\colon \{0,1\}^* \to \{0,1\}^h$ is chosen at random by games Priv and Rec.

SECURITY. Since an encryption key is used by the share algorithm to encrypt just one message, it is natural to think that HK1 is secure if the encryption scheme satisfies one-query indistinguishability (ind1). But we show that the ind1 condition does *not* guarantee privacy of HK1, even in the random-oracle model. Specifically, we show that even one-time-pad encryption, which is certainly ind1-secure, is not enough. Intuitively, the problem is that the hash function is deterministic —even a random oracle is deterministic in the sense that, when invoked twice on the same input, it returns the same answer both times— and hence the values $\boldsymbol{H}[i]$ computed at line 14 can provide partial information about the key $K$.

## 4.2 An attack

We now detail the attack. For concreteness, assume we have $n = 3$ players and wish to use the 2-out-of-3 threshold scheme, access structure $\mathcal{A}_{2,3}$. Assume the domain of secrets is $\mathbb{S} = \{0,1\}^{128}$ and the domain of messages is the same. In the RO-based construction $\text{HK1}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}]$ assume we instantiate $\Pi^{\mathit{Enc}}$ with one-time-pad encryption, $C = \mathit{Encrypt}_K(X) = K \oplus X$. Assume we instantiate $\Pi^{\mathit{PSS}}$

with the 2-out-of-3 Shamir secret-sharing scheme over the finite field $\mathbb{F}_{2^{128}}$. Assume we instantiate $\Pi^{IDA}$ with replication, so $\mathit{Share}^{IDA}(C) = (C, C, C)$. Assume we likewise instantiate $\Pi^{ECC}$ with replication, so $\mathit{Share}^{ECC}(H) = (H, H, H)$.

To understand the attack we first point out that with Shamir's secret-sharing scheme [43], not only can you reconstruct the key (the secret) from $m = 2$ out of $n = 3$ shares, but you can also reconstruct a share (say share 2) given one share (say share 1) and the underlying key $K$ that was dealt. (This is done by interpolation, in the same manner that the secret is normally recovered.) Specifically, for the 2-out-of-3 scheme there is an algorithm $R$ such that $R(\boldsymbol{K}[1], K) = \boldsymbol{K}[2]$ for all $\boldsymbol{K} \in \mathit{Share}^{PSS}(K)$. We will use this fact to violate privacy. Our adversary $A$ selects any two distinct 128-bit strings, $X^0$ and $X^1$, and calls $\mathit{Deal}(X^0, X^1)$. Let $b, K, \boldsymbol{K}, \boldsymbol{C}, \boldsymbol{H}$, and $\boldsymbol{X}$ be as specified in game Priv in response to the *Deal* query. Next, adversary $A$ calls $Corrupt(1)$ to get back $\boldsymbol{X}[1]$, from which it parses out $\boldsymbol{K}[1]$ and $\boldsymbol{C}[1] = C$, the latter because the IDA is replication. It now sets $K^0 = C \oplus X^0$ and $K^1 = C \oplus X^1$. Note that $K^b = K$. Adversary $A$ now defines the candidate share $\boldsymbol{K}^0[2] = R(\boldsymbol{K}[1], K^0)$ for $K^0$ and defines the candidate share $\boldsymbol{K}^1[2] = R(\boldsymbol{K}[1], K^1)$ for $K^1$. We know that $\boldsymbol{K}^b[2] = \boldsymbol{K}[2]$. The adversary $A$ computes $\boldsymbol{H}^0[2] = \mathit{Hash}(\boldsymbol{K}^0[2]\ C)$ and $\boldsymbol{H}^1[2] = \mathit{Hash}(\boldsymbol{K}^1[2]\ C)$. We know that $\boldsymbol{H}^b[2] = \boldsymbol{H}[2]$. But embedded in $\boldsymbol{X}[1]$ is $\boldsymbol{H}[2]$, since the ECC also was replication, which $A$ extracts. So let $A$ return 1 if $\boldsymbol{H}^1[2] = \boldsymbol{H}[2]$ and 0 otherwise. We now show that $A$ has advantage $1 - 2^{-h}$ (recall that $h$ is output length of *Hash*). If $b = 1$ then $A$ always returns 1. From $\boldsymbol{K}[1], \boldsymbol{K}^0[2]$ one can recover $K^0$ and, similarly, from $\boldsymbol{K}[1], \boldsymbol{K}^1[2]$ one can recover $K^1$. But $K^0 \neq K^1$ because $X^0 \neq X^1$, so it must be that $\boldsymbol{K}^0[2] \neq \boldsymbol{K}^1[2]$. We conclude that $\boldsymbol{K}^0[2]\, C \neq \boldsymbol{K}^1[2]$ and so the probability that their hashes collide (under the random-oracle modeled hash-function *Hash*) is at most $2^{-h}$. So if $b = 0$ adversary $A$ outputs 1 with probability $2^{-h}$.

One might be tempted to reason that if the HK1 construction is wrong *even* with a one-time pad and *even* in the RO model, then certainly it is wrong when any "real" encryption scheme and hash-function are used, as these will have inferior properties. But this is not the case, as there are ways in which a "real" encryption scheme is superior to a one-time pad that are of relevance here. The attack above used the fact that with a one-time-pad, given a plaintext/ciphertext pair $(X, C)$ one can recover the key $K$ via $K = C \oplus X$. Had the encryption scheme been secure against one-query key-recovery (key1), meaning that it was computationally infeasible to find the key from a plaintext/ciphertext pair, we would not have been able to mount the attack. And common encryption schemes like CBC mode *do* provide security against key recoverability under standard assumptions.

DISCUSSION. The intent of HK1 was to make shares shorter than the secret. This will not happen if one-time-pad encryption is used, leading one to question the practical relevance of the above counterexample and to ask if ind1 security suffices for encryption schemes in which the ratio of message length to key length is always large. We have not been able to resolve the latter question, and, in particular, have found neither a proof nor a counterexample for whether ind1 implies key1 for encryption schemes of the type just mentioned. As for practical relevance, note that a distributed file system should allow the sharing of files of any length, small or large, so security must be provided even for messages shorter than the key. A reasonable encryption scheme could use one-time-pad encryption for short messages and some other form of encryption for longer ones. Indeed, this could be particularly efficient.

### 4.3 Privacy (in the RO model)

We now show that ind1 + key1 security is enough to prove the security of HK1, in the RO model, under certain conditions on the access structure. Our result applies to threshold access structures or any other adversary class $\mathcal{A}$ where $\mathcal{A} \cap \mathsf{Priv} = \mathcal{A}^{\mathrm{p}}_{m,n}$. This includes $\mathcal{A}_{m,n,t}$ as the distinction between $\mathcal{A}_{m,n,t}$ and $\mathcal{A}_{m,n}$ vanishes after intersecting with $\mathsf{Priv}$.

**Theorem 1 [Privacy of HK1, random-oracle model, threshold schemes]** Let $\mathcal{A} = \mathcal{A}^{\mathrm{p}}_{m,n}$ and let $\Pi = \mathrm{HK1}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}]$ with primitives over $\mathcal{A}$, for $n$-parties, and with $h$-bit hashes. Let $A \in \mathcal{A}$ be an adversary that makes at most $q$ queries to its *Hash* oracle. Then there are adversaries $B_1$ and $B_2$ attacking the symmetric encryption scheme $\Pi^{Enc}$ such that

$$\mathbf{Adv}_{\Pi}^{\text{priv}}(A) \;\leq\; \mathbf{Adv}_{\Pi^{Enc}}^{\text{ind}}(B_1) + 2qn \cdot \mathbf{Adv}_{\Pi^{Enc}}^{\text{key}}(B_2) + \frac{2q+n^2}{2^h}$$

where adversary $B_1$ makes only one query to its left-or-right oracle, adversary $B_2$ makes only one query to its encryption oracle, and the running times of $B_1$ and $B_2$ are that of $A$ plus overhead consisting of one execution of the *Share* algorithm of $\Pi$ and, for $B_2$, an additional $n$ executions of the *Recover* algorithm of $\Pi^{PSS}$. ∎

It is easy to show that $\mathbf{Adv}_{\Pi^{Enc}}^{\text{ind}}(B_1)$ and $\mathbf{Adv}_{\Pi^{Enc}}^{\text{key}}(B_2)$ are small for efficient one-query adversaries $B_1$ and $B_2$ (ind1 + key1 security) if $\mathbf{Adv}_{\Pi^{Enc}}^{\text{ind}}(B_3)$ is small for any efficient two-query adversary (ind2). See Appendix A for a proof. We choose to express our result in terms of ind1 + key1 security in order to precisely hone in on what HK2 needs. Note that a PRP-secure blockcipher is ind1 + key1 secure (even though it is not ind2-secure) and therefore an appropriate realization of $\Pi^{Enc}$ for HK1. Similarly, common modes of operation like CBC are ind1 + key1 secure, even for a fixed IV.

**Proof intuition:** The proof is challenging due to the basic weakness in HK1 exploited in our earlier attack: that the hash function is deterministic and thus may not preserve privacy of the shares to which it is applied. The full proof, which relies on some lemmas concerning PSS privacy from Appendix C.1, is given in Appendix D.

We begin by highlighting two features of the proof. The first is that it relies not just on the privacy but also the recoverability of $\Pi^{PSS}$. (At first glance it is unclear why the privacy of $\Pi$ should need the recoverability of $\Pi^{PSS}$.) The second is that it requires a condition on $\Pi^{PSS}$ that we call *share unpredictability*. This condition is not true for an arbitrary access structure. But it is true for threshold access structures and, more generally, for all access structures that are *extensible*. We define the latter property in Appendix D.

Suppose we aim to construct an adversary $B_1$ attacking the ind1-property of $\Pi^{Enc}$. It would run $A$. The difficulty is that $B_1$ would not know the key $K$ and thus it would be unable to reply to oracle queries of $A$ because these replies are a function of the shares of $K$. We can, however, consider a new game where the plaintext is encrypted under $K$ but the share vector $\boldsymbol{K}$ is produced from a different key $K'$, expecting this to be perfectly adversarially indistinguishable from the original game due to the privacy of the PSS scheme. It is the determinism of the hash function that causes difficulties in establishing something like this. The problem is in answering a hash query of $A$ that contains the share $\boldsymbol{K}[i]$ of an uncorrupted player $i$. This is addressed in two steps. The first is to argue that as long as $m-2$ or fewer players have been corrupted, the share of an uncorrupted player is unpredictable and thus has low probability of being a *Hash* query of $A$. This is true because of the share-unpredictability lemmas, which say that even an adversary knowing the secret and $m-2$ or fewer shares cannot predict any remaining share with reasonable advantage. Here the threshold is $m$, meaning privacy of the secret is guaranteed even if the adversary knows $m-1$ shares, but share-unpredictability allows the adversary only $m-2$ shares, because we need to assume it might also know the secret. The second step is to argue that if the adversary has corrupted $m-1$ players then, if it queries *Hash* on the share of an uncorrupted player, we have $m$ shares of the secret and, via the *Recover* procedure of the PSS scheme, can recover the underlying key. This leads to a key-recovery adversary.

We warn that this sketch elides many issues; see Appendix D.

MINIMALITY OF THE ASSUMPTION. Theorem 1 shows that ind1+key1 security of the encryption scheme is *sufficient* for the privacy of HK1. We now show that it is also *necessary*. That is, we show that for *any* encryption scheme $\Pi^{Enc}$ that is *not* ind1+key1 secure, $\Pi = \text{HK1}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}]$ can fail to provide privacy. The proof of the following is in Appendix G.

**Theorem 2 [Minimality of the ind1+key1 assumption for proving the security of HK1]** Fix an encryption scheme $\Pi^{ECC} = (\mathit{Encrypt}, \mathit{Decrypt})$ and a number $h$. Then there exists $m$, $n$, $\mathcal{A} = \mathcal{A}_{m,n}$, $\Pi^{PSS}$, $\Pi^{IDA}$, and $\Pi^{ECC}$ where, letting $\Pi = \text{HK1}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}]$ (with primitives over $\mathcal{A}$, for $n$-parties, and $h$-bit hashes), for any adversary $B$ there is an adversary $A$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \ \geq \ \mathbf{Adv}^{\mathrm{key}}_{\Pi^{Enc}}(B) - 2^{-h} \ , \tag{1}$$

and for any adversary $B$ there is an adversary $A$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \ \geq \ \mathbf{Adv}^{\mathrm{ind}}_{\Pi^{Enc}}(B) \ . \tag{2}$$

In both cases the running time $A$ is essentially that of $B$ (see the proof), and $A$ makes at most one query to each of its oracles. ▮

Theorems establishing the necessity of an assumption within some protocol are not common, so let us explain why the theorem above accomplishes this. Suppose you wanted to prove that $\Pi = \mathrm{HK1}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}]$ achieved the privacy property assuming that $\Pi^{PSS}$, $\Pi^{IDA}$, and $\Pi^{ECC}$ are good PSS, IDA, and ECC schemes. The theorem above establishes that, if you make an assumption on $\Pi^{Enc}$ that doesn't imply ind1+key1 security, you won't be able to get a proof.

### 4.4 Recoverability (in the RO model)

We prove recoverability for any (valid) class of adversaries, which includes the adversaries associated to any access structure, and $\mathcal{A}_{m,n,t}$ as well. Appendix E.

**Theorem 3 [Recoverability of HK1, random-oracle model]** Let $\mathcal{A}$ be a valid class of adversaries and let $\Pi = \mathrm{HK1}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}]$ with primitives over $\mathcal{A}$, for $n$ parties, and with $h$-bit hashes. Let $A \in \mathcal{A}$ be an adversary that asks at most $q$ queries to its *Hash* oracle. Then $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) \ \leq \ (q+2n)^2/2^{h+1}$ . ▮

The recoverability of HK1 requires only the collision-intractability of the hash function *Hash*; it is possible to restate the theorem above and adjust its proof to show that an attack on the recoverability of HK1 implies an equally effective method to find collisions in *Hash*. We didn't express the result this way since the proof of privacy was already in the random-oracle model.

## 5 The HK2 Protocol (Refining Krawczyk's Scheme)

We now alter HK1 by replacing its deterministic hash function *Hash* with a randomized commitment scheme. This changes the protocol, as the randomness used in the commitment must be inserted into the shares. We are then able to show that the new protocol, HK2, is a good RCSS under standard assumptions.

### 5.1 The construction

COMMITMENT SCHEMES. We formalize a (noninteractive) commitment scheme as a pair $\Pi^{Com} = (\mathit{Ct}, \mathit{Vf})$. Here *Ct* is a probabilistic algorithm that takes a message $M \in \{0,1\}^*$ and returns either a pair $(Y, R)$, where $Y$ is the *committal* and $R$ is the *decommittal*, or else it returns $\perp$. Algorithm *Vf* is deterministic and, on input $Y, M, R$, returns a bit. The *domain* $\mathrm{Dom} \subseteq \{0,1\}^*$ of $\Pi^{Com}$ is the set of all $M \in \{0,1\}^*$ such that $\mathit{Ct}(M)$ is never $\perp$. We assume that whether or $\mathit{Ct}(M)$ is $\perp$ is independent of its coin tosses (which ensures that it is easy to check if a point is in the domain).

There are two security properties, *hiding* and *binding*, each defined by a game. See Figure 6. In game Hide, multiple queries to *LeftOrRight* are allowed, and arguments $M_0$ and $M_1$ to *LeftOrRight* need not be of equal length. The advantage of $A$ in attacking the hiding-property of the commitment scheme is $\mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(A) = 2\Pr[\mathrm{Hide}^A] - 1$. We say that $\Pi^{Com}$ is $\epsilon(\cdot)$-hiding if $\mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(A) \leq \epsilon(q)$ for any adversary $A$ that makes at most $q$ oracle queries. Note that the adversary is not computationally restricted; we have given a statistical

| | |
|---|---|
| PROCEDURE *Initialize*<br>$b \xleftarrow{\$} \{0,1\}$<br><br>PROCEDURE *Finalize* $(d)$<br>RETURN $b = d$ | PROCEDURE *LeftOrRight*$(M_0, M_1)$     Game Hide<br>IF $M_0 \notin \mathrm{Dom}$ OR $M_1 \notin \mathrm{Dom}$<br>  THEN RETURN $\perp$<br>$(Y, R) \xleftarrow{\$} \mathit{Ct}(M_b)$<br>RETURN $Y$ |
| PROCEDURE *Commit*$(M_0)$<br>IF $M_0 \notin \mathrm{Dom}$ THEN RETURN $\perp$<br>$(Y, R_0) \xleftarrow{\$} \mathit{Ct}(M)$<br>RETURN $(Y, R_0)$ | PROCEDURE *Finalize* $(M_1, R_1)$     Game Bind<br>IF $M_1 \notin \mathrm{Dom}$ THEN RETURN $\perp$<br>RETURN $(M_0 \neq M_1$ AND<br> $\mathit{Vf}(Y, M_0, R_0) = 1$ AND<br> $\mathit{Vf}(Y, M_1, R_1) = 1)$ |

Figure 6: Games used to define the security of a commitment scheme $\Pi^{\mathit{Com}} = (\mathit{Ct}, \mathit{Vf})$ with domain $\mathrm{Dom}$.

| | |
|---|---|
| PROCEDURE *Share*$(X)$<br>$K \xleftarrow{\$} \{0,1\}^k$ ; $C \xleftarrow{\$} \mathit{Encrypt}_K(X)$<br>$\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(K)$ ; $\boldsymbol{C} \xleftarrow{\$} \mathit{Share}^{\mathit{IDA}}(C)$<br>FOR $i \leftarrow 1$ TO $n$ DO<br>    $(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} \mathit{Ct}(\boldsymbol{K}[i]\,\boldsymbol{C}[i])$<br>    $\boldsymbol{S}_i \xleftarrow{\$} \mathit{Share}^{\mathit{ECC}}(\boldsymbol{H}[i])$<br>FOR $i \leftarrow 1$ TO $n$ DO<br>    $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i]$<br>RETURN $\boldsymbol{X}$ | PROCEDURE *Recover*$(\boldsymbol{X}, j)$<br>FOR $i \leftarrow 1$ TO $n$ DO<br>    $\boldsymbol{R}[i]\boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i] \leftarrow \boldsymbol{X}[i]$<br>FOR $i \leftarrow 1$ TO $n$ DO $\boldsymbol{H}[i] \leftarrow \mathit{Recover}^{\mathit{ECC}}(\boldsymbol{S}_i, j)$<br>FOR $i \leftarrow 1$ TO $n$ DO<br>    IF $\boldsymbol{X}[i] \neq \Diamond$ AND $\mathit{Vf}(\boldsymbol{H}[i], \boldsymbol{K}[i]\,\boldsymbol{C}[i], \boldsymbol{R}[i]) = 0$<br>        THEN $\boldsymbol{K}[i] \leftarrow \Diamond$ ; $\boldsymbol{C}[i] \leftarrow \Diamond$<br>$K \leftarrow \mathit{Recover}^{\mathit{PSS}}(\boldsymbol{K}, j)$ ; $C \leftarrow \mathit{Recover}^{\mathit{IDA}}(\boldsymbol{C}, j)$<br>$X \leftarrow \mathit{Decrypt}_K(C)$<br>RETURN $X$ |

Figure 7: Definition of the HK2 construction $\Pi = (\mathit{Share}, \mathit{Recover}) = \mathrm{HK2}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \Pi^{\mathit{Com}}]$.

notion of privacy. For the *binding game*, Bind, there is no *Initialize* procedure. We define the advantage of $A$ in attacking the binding-property of the commitment scheme as $\mathbf{Adv}^{\mathrm{bind}}_{\Pi^{\mathit{Com}}}(A) = \Pr[\mathrm{Bind}^A]$. The notion is weaker than the classical notion of binding, which would speak to the computational infeasibility to find any $Y, M_0, R_0, M_1, R_1$ such that $M_0, M_1 \in \mathrm{Dom}$ AND $M_0 \neq M_1$ AND $\mathit{Vf}(Y, M_0, R_0) = 1$ AND $\mathit{Vf}(Y, M_1, R_1) = 1$. The conventional notion is analogous to the collision resistance of a hash function while our notion is more like a UOWHF [37] (also called TCR hash-function [5]). Informally, we refer to a commitment scheme $\Pi^{\mathit{Com}}$ as *statistically-hiding, weakly-binding* (SHWB) if $\mathbf{Adv}^{\mathrm{hide}}_{\Pi}(A)$ is small for any reasonable adversaries $A$ and $\mathbf{Adv}^{\mathrm{bind}}_{\Pi}(A)$ is small for any reasonable adversaries $A$.

THE HK2 SCHEME. Fix an adversary class $\mathcal{A}$. We build an $n$-party secret-sharing scheme with message space $\mathbb{S}$ from components: (1) a symmetric encryption scheme $\Pi^{\mathit{Enc}} = (\mathit{Encrypt}, \mathit{Decrypt})$ with $k$-bit keys and a message space $\mathbb{S}$; (2) an $n$-party PSS $\Pi^{\mathit{PSS}} = (\mathit{Share}^{\mathit{PSS}}, \mathit{Recover}^{\mathit{PSS}})$ over $\mathcal{A}$ with message space $\{0,1\}^k$; (3) an $n$-party IDA $\Pi^{\mathit{IDA}} = (\mathit{Share}^{\mathit{IDA}}, \mathit{Recover}^{\mathit{IDA}})$ over $\mathcal{A}$ with message space $\Sigma^*$; (4) an $n$-party ECC $\Pi^{\mathit{ECC}} = (\mathit{Share}^{\mathit{ECC}}, \mathit{Recover}^{\mathit{ECC}})$ over $\mathcal{A}$ with message space $\{0,1\}^h$; and (5) a commitment scheme $\Pi^{\mathit{Com}} = (\mathit{Ct}, \mathit{Vf})$ with domain $\mathrm{Dom}$ where $\boldsymbol{K}[i]\,\boldsymbol{C}[i] \in \mathrm{Dom}$ if $\boldsymbol{K} \in \mathit{Share}^{\mathit{PSS}}(K)$ and $\boldsymbol{C} \in \mathit{Share}^{\mathit{IDA}}(C)$ for some $K \in \{0,1\}^k$, $X \in \mathbb{S}$, and $C \in \mathit{Encrypt}_K(X)$. We call $\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \Pi^{\mathit{Com}}$ the underlying *primitives* of the HK2 scheme, and we say that they are over $\mathcal{A}$, and for $n$ parties. From such a set of primitives we define the secret-sharing scheme $\mathrm{HK2}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \Pi^{\mathit{Com}}] = (\mathit{Share}, \mathit{Recover})$ as specified by Figure 7. The figure uses the same conventions as those of Figure 5.

PROCEDURE *Initialize* $\quad G_0$–$G_2$

$K \xleftarrow{\$} \{0,1\}^k\ ;\ b \xleftarrow{\$} \{0,1\}\ ;\ \boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(K)$

RETURN

---

PROCEDURE $\mathit{Deal}(X^0, X^1)$ $\quad G_0, G_1, G_4, G_5$

$C \xleftarrow{\$} \mathit{Encrypt}_K(X^b)\ ;\ \boldsymbol{C} \xleftarrow{\$} \mathit{Share}^{\mathit{IDA}}(C)$

FOR $i \leftarrow 1$ TO $n$ DO

$\quad (\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} \mathit{Ct}(\boldsymbol{K}[i]\, C[i])\ ;\ \boldsymbol{S}_i \xleftarrow{\$} \mathit{Share}^{\mathit{ECC}}(\boldsymbol{H}[i])$

---

PROCEDURE $\mathit{Corrupt}(i)$ $\quad G_0, G_5$

$\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\, \boldsymbol{K}[i] C[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i]$

RETURN $\boldsymbol{X}[i]$

---

PROCEDURE $\mathit{Finalize}(d)$ $\quad G_0$–$G_5$

RETURN $(d = b)$

---

PROCEDURE *Initialize* $\quad G_3$–$G_5$

$K, K' \xleftarrow{\$} \{0,1\}^k\ ;\ b \xleftarrow{\$} \{0,1\}$

$\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(K')$

---

PROCEDURE $\mathit{Deal}(X^0, X^1)$ $\quad G_2, G_3$

$C \xleftarrow{\$} \mathit{Encrypt}_K(X^b)\ ;\ \boldsymbol{C} \xleftarrow{\$} \mathit{Share}^{\mathit{IDA}}(C)$

FOR $i \leftarrow 1$ TO $n$ DO

$\quad (\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} \mathit{Ct}(0\, \boldsymbol{C}[i])$

$\quad \boldsymbol{S}_i \xleftarrow{\$} \mathit{Share}^{\mathit{ECC}}(\boldsymbol{H}[i])$

---

PROCEDURE $\mathit{Corrupt}(i)$ $\quad G_1$–$G_4$

$\boldsymbol{R}[i] \xleftarrow{\$} \mathit{DCt}(\boldsymbol{H}[i], \boldsymbol{K}[i]\, \boldsymbol{C}[i])$

$\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\, \boldsymbol{K}[i]\, \boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i]$

RETURN $\boldsymbol{X}[i]$

Figure 8: Games for proving Theorem 4, the privacy of the HK2 scheme.

## 5.2 Privacy (in the standard model)

The difficulty in establishing privacy in the standard model is that our adversary is dynamic, and so we run into the *selective-decommitment problem*; see Dwork, Naor, and Reingold [19]. One could always pretend the adversary to be static and take a hit of $2^n$ in the security bound when the adversary is dynamic, but we don't want to do this, as we are interested in concrete security and results with good asymptotic counterparts. Another way around this is to use a statistically-hiding chameleon commitment-scheme. Instead we make do with a weaker requirement, just the statistical hiding. We comment that for the case of static adversaries it would suffice that the commitment be computationally rather than statistically hiding.

**Theorem 4 [Privacy of HK2]** Let $\mathcal{A}$ be an adversary class and $\Pi = \mathrm{HK2}[\Pi^{\mathit{Enc}}, \Pi^{\mathit{PSS}}, \Pi^{\mathit{IDA}}, \Pi^{\mathit{ECC}}, \Pi^{\mathit{Com}}]$ with primitives over $\mathcal{A}$, for $n$ parties, and with an $\epsilon(\cdot)$-hiding $\Pi^{\mathit{Com}}$. Let $A \in \mathcal{A} \cap \mathsf{Priv}$ be an adversary for attacking the privacy of $\Pi$. Then there is an adversary $B$ for attacking the privacy of $\Pi^{\mathit{Enc}}$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \ \leq\ \mathbf{Adv}^{\mathrm{ind}}_{\Pi^{\mathit{Enc}}}(B) + 4\epsilon(n)$$

where $B$ makes only one query to its left-or-right oracle and the running time of $B$ is that of $A$ plus overhead consisting of one execution of the *Share* algorithm of $\Pi$. ▮

**Proof:** [Theorem 4]The proof relies on the games in Figure 8. The figure shows many procedures, indicating next to each in which games it is included. For example, game $G_0$ is defined by the procedures on the left-hand-side of the figure. The procedure $\mathit{Corrupt}$ of games $G_1$–$G_4$ refers to a probabilistic algorithm *DCt* that works as follows. On input $Y, M$ it lets $\Omega(Y, M)$ denote the set of all coins $\omega$ such that *Ct*, on input $M$ and coins $\omega$, returns a pair whose first component is $Y$. If $\Omega(Y, M) = \emptyset$ then *DCt* returns $\bot$. Else it picks $\omega$ at random from $\Omega(Y, M)$, runs *Ct* on input $M$ and coins $\omega$ to get a pair $(Y, R)$, and returns $R$. Note this algorithm is not necessarily efficiently implementable. We note that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \ =\ 2 \cdot \Pr\left[G_0^A\right] - 1\ . \tag{3}$$

Game $G_1$ differs from game $G_0$ only in the $\mathit{Corrupt}$ procedure, which resamples $\boldsymbol{R}[i]$ as shown. Clearly,

$$\Pr\left[G_0^A\right] = \Pr\left[G_1^A\right] = \Pr\left[G_2^A\right] + (\Pr\left[G_1^A\right] - \Pr\left[G_2^A\right])\ . \tag{4}$$

We will construct an adversary $D_1$ attacking the hiding-property of $\Pi^{Com}$ such that

$$\Pr\left[G_1^A\right] - \Pr\left[G_2^A\right] \;=\; \mathbf{Adv}^{\rm hide}_{\Pi^{Com}}(D_1)\;. \tag{5}$$

Adversary $D_1$ picks $b \xleftarrow{\$} \{0,1\}$ and runs $A$. When $A$ makes a query $X^0, X^1$ to its *Deal* oracle, adversary $D_1$ picks $K \xleftarrow{\$} \{0,1\}^k$ and $C \xleftarrow{\$} \textit{Encrypt}_K(X^b)$. It then picks $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{PSS}(K)$. For $i$ running from 1 to $n$, it queries $0\,\boldsymbol{C}[i], \boldsymbol{K}[i]\,\boldsymbol{C}[i]$ to its *LeftOrRight* oracle, lets $\boldsymbol{H}[i]$ denote the value returned, and lets $\boldsymbol{S}_i \xleftarrow{\$} \textit{Share}^{ECC}(\boldsymbol{H}[i])$. When $A$ makes a $Corrupt(i)$ query, adversary $D_1$ computes its reply according to the code of the $Corrupt$ procedure of games $G_1, G_2$. Note that this step is not necessarily efficient, but $D_1$ does not have to be computationally bounded. When $A$ halts without output $d$, adversary $D$ returns 1 if $d = b$ and 0 otherwise. One can check that (5) is true.

Next we have

$$\Pr\left[G_2^A\right] \;=\; \Pr\left[G_3^A\right] + (\Pr\left[G_2^A\right] - \Pr\left[G_3^A\right])\;, \tag{6}$$

where $G_3$ differs from $G_2$ only in the *Initialize* procedure which now produces $\boldsymbol{K}$ by sharing not $K$ but an independently and randomly chosen key $K'$. We claim that

$$\Pr\left[G_2^A\right] \;=\; \Pr\left[G_3^A\right]\;. \tag{7}$$

To justify the above, we build an adversary $P$ attacking the privacy of the PSS scheme $\Pi^{PSS}$ such that

$$\mathbf{Adv}^{\rm priv}_{\Pi^{PSS}}(P) \;=\; \Pr\left[G_2^A\right] - \Pr\left[G_3^A\right]\;. \tag{8}$$

But the privacy of $\Pi^{PSS}$ tells us that the advantage of $P$ is zero, yielding (7). Adversary $P$ begins by picking $K$ and $K'$ at random from $\{0,1\}^k$ and $b$ at random from $\{0,1\}$. It then queries $K', K$ to its *Deal* oracle. We know that the latter creates shares $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{PSS}(L)$ where $L = K'$ if the challenge bit chosen by game Priv is zero and $L = K$ if it is one. Now $P$ starts running $A$, responding to $A$'s oracle queries as follows. When $A$ queries *Deal*$(X^0, X^1)$ adversary $P$ executes the code of the *Deal* procedure of games $G_2, G_3$. When $A$ makes a $Corrupt(i)$ query, adversary $P$ itself makes a $Corrupt(i)$ query to obtain share $\boldsymbol{K}[i]$, produces $\boldsymbol{X}[i]$ as per the code of the $Corrupt$ procedure of games $G_2, G_3$, and returns $\boldsymbol{X}[i]$ to $A$. As before, this step is not necessarily efficient, but $P$ need not be computationally bounded. When $A$ halts and outputs a bit $d$, adversary $P$ returns 1 if $b = d$ and 0 otherwise. It is easy to see that (8) is true.

Next we have

$$\Pr\left[G_3^A\right] \;=\; \Pr\left[G_4^A\right] + (\Pr\left[G_3^A\right] - \Pr\left[G_4^A\right])\;. \tag{9}$$

We next construct an adversary $D_2$ attacking the hiding-property of $\Pi^{Com}$ such that

$$\Pr\left[G_3^A\right] - \Pr\left[G_4^A\right] \;=\; \mathbf{Adv}^{\rm hide}_{\Pi^{Com}}(D_2)\;. \tag{10}$$

The construction of $D_2$ is similar to that of $D_1$ and is therefore omitted. Games $G_5$ differs from $G_4$ only in its $Corrupt$ procedure as shown. Clearly

$$\Pr\left[G_4^A\right] \;=\; \Pr\left[G_5^A\right]\;. \tag{11}$$

We now construct adversary $B$ attacking the privacy of $\Pi^{Enc}$ such that

$$2 \cdot \Pr\left[G_5^A\right] - 1 \;\leq\; \mathbf{Adv}^{\rm ind}_{\Pi^{Enc}}(B)\;. \tag{12}$$

Adversary $B$ picks $K'$ at random and lets $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{PSS}(K')$. It then runs $A$. When $A$ makes a query *Deal*$(X^0, X^1)$, $B$ queries $X^0, X^1$ to its own left-or-right encryption oracle to get back $C \xleftarrow{\$} \textit{Encrypt}_K(X^b)$, where $b$ is the challenge bit and $K$ the key chosen by the Ind game defining the privacy of $\Pi^{Enc}$. Now $B$ executes the last five lines of the *Deal* procedure of game $G_5$. When $A$ makes a $Corrupt(i)$ query, adversary $B$

can execute the code of the $Corrupt$ procedure of game $G_5$ since it knows $\boldsymbol{K}[i]$. When $A$ halts and outputs a bit $d$, adversary $B$ returns $d$. The advantage of $B$ is $2\Pr[b = d] - 1$, so (12) is true.

Let $D$ be the hiding-adversary that flips a fair coin and, if it lands heads, runs $D_1$, otherwise, $D_2$. Clearly

$$\mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(D) \;=\; 0.5 \cdot \mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(D_1) + 0.5 \cdot \mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(D_2)\;. \tag{13}$$

Since $\Pi^{Com}$ is assumed to be $\epsilon(\cdot)$-hiding and $D$ makes at most $n$ oracle queries we have

$$\mathbf{Adv}^{\mathrm{hide}}_{\Pi^{Com}}(D) \;\leq\; \epsilon(n)\;. \tag{14}$$

Putting together (3)–(14) concludes the proof. ▌

### 5.3 Recoverability (in the standard model)

We now establish the recoverability of HK2. The theorem applies to any valid adversary class and assumes a weakly-binding committal. The proof is in Appendix F.

**Theorem 5 [Recoverability of HK2]** Let $\mathcal{A}$ be a valid adversary class and let $\Pi = \mathrm{HK2}[\Pi^{Enc}, \Pi^{PSS}, \Pi^{IDA}, \Pi^{ECC}, \Pi^{Com}]$ with primitives over $\mathcal{A}$ and for $n$ parties. Let $A \in \mathcal{A}$. Then there is an adversary $B$ attacking the binding-property of $\Pi^{Com}$ such that $\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) \leq n \cdot \mathbf{Adv}^{\mathrm{bind}}_{\Pi^{Com}}(B)$ and where the running time of $B$ is that of $A$ plus overhead consisting of an execution of the *Share* and *Recover* algorithms of protocol $\Pi$. ▌

REALIZING THE COMMITMENT. Constructions are known for noninteractive, statistically-hiding commitment-schemes that meet the standard binding requirement, and therefore our own. One is based on discrete log [11], another, on a collision-resistant hash-function [18, 23]. These constructions are all reasonably efficient. Actually, having relaxed the binding requirement, one can replace the collision-resistant hash-function of the constructions just mentioned with the UOWHF primitive of Naor and Yung [37]. This provides a basis for the plausibility-style result that a one-way function suffices for efficient RCSS [42][6], and it also provides the basis for a practical scheme that builds its UOWHF from appropriately keying a cryptographic hash-function.

## Acknowledgments

Many thanks to Mark O'Hare and Rick Orsini, of Security First Corp., for calling our attention to the foundational issues of RCSS and for supporting our work to resolve them. Thanks to Hugo Krawczyk for helpful comments.

## References

[1] P. Béguin and A. Cresti. General short computational secret sharing schemes. *Advances in Cryptology – EUROCRYPT '95*, LNCS vol. 921, pp. 194–208, 1995.

[2] A. Beimel and B. Chor. Universally ideal secret sharing schemes. *IEEE Trans. on Info. Theory*, vol. 40, no. 3, pp. 786–794, 1994.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. *38th Annual Symposium on Foundations of Computer Science* (FOCS 1997), pp. 394–403, 1997.

[6] Statistically hiding commitment-schemes satisfying the standard (rather than our weakened) notion of binding can be built from one-way permutations [36] and even one-way functions [24]. But these schemes are interactive, and so unsuitable for our application.

[4] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. *Advances in Cryptology – EUROCRYPT '06*, LNCS vol. 4004, Springer, pp. 409–426, 2006.

[5] M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. *Advances in Cryptology – CRYPTO '97*, LNCS vol. 1294, Springer, pp. 470–484, 1997.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proc. of the First Annual Conference on Computer and Communications Security* (ACM CCS), ACM Press, 1993.

[7] M. Bellare and P. Rogaway. Robust computational secret sharing and a unified account of classical secret-sharing goals. Proceedings version of this paper. *Proc. of the 14th ACM Conference on Computer and Communications Security* (ACM CCS), ACM Press, 2007.

[8] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. *Advances in Cryptology – CRYPTO '88*, LNCS vol. 403, Springer, pp. 27–36, 1990.

[9] G. Blakley. Safeguarding cryptographic keys. *AFIPS National Computer Conference*, vol. 48, pp. 313–317, 1979.

[10] C. Blundo, A. De Santis, G. Di Crescenzo, A. Gaggia, and U. Vaccaro. Multi-secret sharing schemes. *Advances in Cryptology – CRYPTO '94*, LNCS vol. 839, Springer, pp. 150–163, 1994.

[11] J. Boyar, S. Kurtz, and M. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *J. of Cryptology*, vol. 2, no. 2, pp. 63–76, 1990.

[12] E. Brickell and D. Stinson. The detection of cheaters in threshold schemes. *SIAM J. of Discrete Math*, vol. 4, no. 4, pp. 502–510, 1991.Earlier version in Crypto 88.

[13] E. Brickell and D. Stinson. Some improved bounds on the information rate of perfect secret sharing schemes. *J. of Cryptology*, vol. 5, 153–166, 1992.

[14] C. Cachin. On-line secret sharing. *IMA Conference on Cryptography and Coding*, LNCS vol. 1025, Springer, pp. 190-198, 1995.

[15] R. Capocelli, A. DeSantis, L. Gargano, and U. Vaccaro. On the size of shares for secret sharing schemes. *J. of Cryptology*, vol. 6, pp. 157–167, 1993.

[16] M. Carpentieri, A. De Santis, and U. Vaccaro. Size of shares and probability of cheating in threshold schemes. *Advances in Cryptology – EUROCRYPT '93*, LNCS vol. 765, Springer, pp. 117–125, 1993.

[17] B. Chor, S. Goldwasser, S. Micali, and B. Awerbach. Verifiable secret sharing and achieving simultaneity in the presence of faults. *FOCS '85*, IEEE Press, pp. 383–395, 1985.

[18] I. Damgård, T. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. of Cryptology*, vol. 10, no.3, pp. 163–194, 1997.

[19] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *JACM*, vol. 50, no. 6, pp. 852–921, 2003.

[20] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *FOCS '87*, IEEE Computer Society, pp. 427–437, 1987.

[21] G. Ganger, P. Khosla, M. Bakkaloglu, M. Bigrigg, G. Goodson, S. Oguz, V. Pandurangan, C. Soules, J. Strunk, and J. Wylie. Survivable storage systems. *DARPA Information Survivability Conference and Exposition*, vol. 2, IEEE Press, pp. 184–195, 2001.

[22] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences* (JCSS), vol. 28, no. 2, pp. 270–299, 1984.

[23] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. *Advances in Cryptology – CRYPTO '96*, LNCS vol. 1109, Springer, pp. 201-215, 1996.

[24] I. Haitner and O. Reingold. Statistically-hiding commitment from any one-way function. Cryptology ePrint report 2006/436, 2006.

[25] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: how to cope with perpetual leakage. *Advances in Cryptology – CRYPTO '95*, LNCS vol. 963, Springer, pp. 339–352, 1998.

[26] Y. Ishai. Personal communication, February 2007.

[27] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structure. *IEEE Globecom 87*, pp. 99–102, 1987.

[28] A. Iyengar, R. Cahn, C. Jutla, and J. Garay. Design and implementation of a secure distributed data repository. *14th IFIP International Information Security Conference*, pp. 123–135, 1998.

[29] W. Jackson and K. Martin. Combinatorial models for perfect secret-sharing schemes. *J. of Comb. Mathematics and Comb. Computing*, vol. 28, pp. 249–265, 1998.

[30] E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory,* vol. 29, no. 1, pp. 35–51, 1983.

[31] H. Krawczyk. Secret sharing made short. LNCS vol. 773, Springer, pp. 136–146, 1993. *Advances in Cryptology – CRYPTO '93.*

[32] H. Krawczyk. Distributed fingerprints and secure information dispersal. *Twelfth Annual ACM Symposium on Principles of Distributed Computing* (PODC 1993), ACM Press, pp. 207–218, 1993.

[33] S. Lakshmanan, M. Ahamad, and H. Venkateswaran. Responsive security for stored data. *IEEE Trans. on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 818–828, 2003.

[34] A. Mayer and M. Yung. Generalized secret sharing and group-key distribution using short keys. *Compression and Complexity of Sequences 1997*, IEEE Press, pp. 30–44, 1997.

[35] R. McEliece and D. Sarwate. On sharing secrets and Reed-Solomon codes. *Communication of the ACM*, vol. 24, pp. 583–584, 1981.

[36] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, vol. 11, no. 2, pp. 87–108, 1998.

[37] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *Twenty first Annual ACM Symposium on Theory of Computing* (STOC 1989), IEEE Press, pp. 33–43, 1989.

[38] W. Ogata, K. Kurosawa, and D. Stinson. Optimum secret sharing scheme secure against cheating. *SIAM J. on Discreet Mathematics*, vol. 20, no. 1, pp. 79–95, 2006.

[39] A. Paul, S. Adhikari, and U. Ramachandran. Design of a secure and fault tolerant environment for distributed storage. Georgia Tech Center for Experimental Research in Computer Science (CERCS) Technical Report GIT-CERCS-04-02, 2004.

[40] M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[41] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. *Symposium on the Theory of Computing* (STOC 1989), ACM Press, pp. 730-85, 1989.

[42] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *STOC '90*, pp. 387–394, 1990.

[43] A. Shamir. How to share a secret. *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[44] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, vol. 27, pp. 379–423 and pp. 623–656, July and October, 1948.

[45] G. Simmons. How to (really) share a secret. *Advances in Cryptology – CRYPTO '88*, LNCS vol. 403, Springer, pp. 390–448, 1989.

[46] G. Simmons. An introduction to shared secret and/or shared control schemes and their application. Chapter 9 from *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, pp. 441–497, 1991.

[47] M. Stadler. Publicly verifiable secret sharing. *Advances in Cryptology – EUROCRYPT '96.* LNCS vol. 1070, Springer, pp. 190–199, 1996.

[48] D. Stinson. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, vol. 2, Kluwer, pp. 357–390, 1992.

[49] D. Stinson and R. Wei. Bibliography of secret sharing schemes. On-line bibliography, 216 references, dated 13 Oct 1998. http://www.cacr.math.uwaterloo.ca/ dstinson/ssbib.html.

[50] M. Tompa and H. Woll. How to share a secret with cheaters. *Journal of Cryptology*, vol. 1, pp. 133–138, 1988. Earlier version in *Crypto '86.*

[51] V. Vinod, A. Narayanan, K. Srinathan, C. Rangan, and K. Kim. On the power of computational secret sharing. *Progress in Cryptology – INDOCRYPT 2003*, LNCS vol. 2904, Springer, pp. 162–176, 2003.

[52] M. Waldman, A. Rubin, and L. Cranor. The architecture of robust publishing systems. *ACM Transactions on Internet Technology* (TOIT), vol. 1, no. 2, pp. 199–230, 2001.

[53] H. Witsenhausen. The zero-error side information problem and chromatic numbers. *IEEE Transactions on Information Theory*, vol. 22, no. 5, pp. 592–593, 1976.

[54] J. Wylie, M. Bigrigg, J. Strunk, G. Ganger, H. Kiliççöte, and P. Khosla. Survivable information storage systems. *IEEE Computer*, vol. 33, no. 8, pp. 61–68, August 2000.

# A A Sufficient Condition for key1-Security

An encryption scheme secure against $q \geq 2$ queries in the indistinguishability sense is also secure against $q - 1$ queries in key-recoverability sense (so, in particular, ind2-security implies key1-security). For completeness, we formalize and prove this below. In particular, two-query indistinguishability $(\mathrm{ind2})$ implies one-query key-recoverability (key1), but an encryption scheme secure in the key1 sense need not be secure against key-recovery at all (the one-time pad is an example).

**Proposition 6** Let $\Pi = (\mathit{Encrypt}, \mathit{Decrypt})$ be an encryption scheme with message space including $\{0,1\}^m$ for some $m$. Let $A$ be a (key-recovery) adversary. Then there exists a (distinguishing) adversary $D$ such that $\mathbf{Adv}_\Pi^{\mathrm{ind}}(D) \geq \mathbf{Adv}_\Pi^{\mathrm{key}}(A) - 2^{-m}$ and where $D$ makes one more oracle query than does $A$, makes oracle queries of total length $m$ bits more than the total length of $A$'s queries, and $D$ runs in time which is $A$'s running time plus the time for one *Decrypt* call on an $m$-bit string. ∎

**Proof:** Construct $D$ as follows. It runs $A$, answering each $\mathit{Enc}(X)$ query by calling $\mathit{LeftOrRight}(X, X)$ and returning the response. When $A$ halts with output $K'$, have $D$ compute $X \xleftarrow{\$} \{0,1\}^m$, $C \xleftarrow{\$} \mathit{LeftOrRight}(X, 0^m)$, and $X' = \mathit{Decrypt}_{K'}(C)$. Let $D$ return 0 if $X = X'$ and 1 otherwise.

Let Left and Right denote the games that are the same as the Ind game except the encryption oracle *Enc* is replaced by the oracle that always encrypts the left or right queries, respectively. Suppose that $D$ plays game Left. Then the probability that $D$ will output `true` is at least $\mathbf{Adv}_\Pi^{\mathrm{key}}(A)$. On the other hand, suppose that $D$ plays game Right. Then if $D$ outputs `true` it means that $D$, given *no* information about $X$, managed to correctly guess it. The chance of this is at most $2^{-m}$. Now, as is standard, $\mathbf{Adv}_\Pi^{\mathrm{ind}}(D) = 2\Pr[\mathrm{Ind}^D] - 1 = \Pr[\mathrm{Left}^D \Rightarrow \mathtt{true}] - \Pr[\mathrm{Right}^D]$, and so we conclude that $\mathbf{Adv}_\Pi^{\mathrm{ind}}(D) \geq \mathbf{Adv}_\Pi^{\mathrm{key}}(A) - 2^{-m}$. ∎

While ind2-security implies ind1+key1 security, the reverse certainly is not the case. As an example, CBC encryption with a zero-IV is readily shown to be ind1+key1 secure (when based on a PRP), but it is not ind2 secure. It is for this reason that Theorem 1 employs the weaker ind1+key1 assumption.

# B Prior Secret-Sharing Definitions

The purpose of this section is to sketch the most prominent definitions for classical secret-sharing goals. We do not aim to give a comprehensive survey, which would include many variations of the same.

BLAKLEY AND SHAMIR (1979). A threshold scheme with parameters $m$ and $n$ (that is, a secret-sharing scheme for the access structure $\mathcal{A}_{m,n}$) was defined by Shamir [43] as follows[7]: *Our goal is to divide $S$ into $n$ pieces $S_1, \ldots, S_n$ in such a way that: (1) knowledge of any $m$ or more $S_i$ pieces makes $S$ easily computable; and (2) knowledge of any $m-1$ or fewer $S_i$ pieces leaves $S$ completely undetermined (in the sense that all its possible values are equally likely).*

The definition above is somewhat informal, and admits multiple, basically equivalent formalizations. The two most prominent are the *conditional-probability formulation* and the *entropy formulation*. One can either assume that the finite set of possible secrets $\mathbb{S}$ is endowed with a distribution and define a threshold scheme *for* this distribution, or one can require the scheme to work for *any* distribution $\mathbb{S}$; see, for example, [2, 29]. Illustrating the former approach, let $S$ denote the random variable that takes on values from $\mathbb{S}$ according to the associated distribution and let $S_i$ be the random variable that takes on values of the share $i$ for $i \in [n]$. For the conditional-probability formulation one would then require that for any distinct $\{i_1, \ldots, i_r\} \subseteq [n]$ and any $(s_{i_1}, \ldots, s_{i_r})$ such that $\Pr[(S_{i_1}, \ldots, S_{i_r}) = (s_{i_1}, \ldots, s_{i_r})] > 0$, we have that: (1) if $r \geq m$ then there exists a unique $s \in \mathbb{S}$ such that $\Pr[S = s \mid S_{i_1} = s_{i_1} \wedge \cdots \wedge S_{i_r} = s_{i_r}] = 1$; and (2) if $r < m$ then, for each $s \in \mathbb{S}$ we have that $\Pr[S = s \mid S_{i_1} = s_{i_1} \wedge \cdots \wedge S_{i_r} = s_{i_r}] = \Pr[S = s]$. The statement we have just given paraphrases [38]. For the entropy formalization [30] one would require that: (1) for any $m$-tuple of distinct indices $i_1, \ldots, i_m \in [n]$ we have that $H(S \mid S_{i_1}, \ldots, S_{i_m}) = 0$; and (2) for any $r < m$ and for any $r$-tuple of distinct indices $i_1, \ldots, i_r \in [n]$ we have that $H(S \mid S_1, \ldots, S_r) = H(S)$. Here $H(X) = -\sum_{x \in X} p(x) \lg p(x)$ and $H(X \mid Y) = -\sum_{x \in X, y \in Y} p(x) p(x \mid y) \lg p(x \mid y)$ and $X$ and $Y$ are random variables and $p(x)$ denotes the probability that $X = x$ and $p(y)$ denotes the probability that $Y = y$ and $p(x \mid y)$ denotes the probability that $X = x$ given that $Y = y$. Both formulations of the PSS notion readily lift to define secret-sharing schemes over an arbitrary access structure $\mathcal{A}$.

[7] For consistency with the rest of this paper, we have changed the names of variables.

MCELIECE AND SARWATE (1981). These authors were interested in threshold schemes that are secure against computationally-unbounded adversaries that can arbitrarily replace the shares of some $t$ of the players [35]. An external party, not a protocol participant, recovers the secret. It is not possible to say precisely what notion the authors aim for because their work is stated in terms of characteristics of schemes achievable using Reed-Solomon codes, not general characteristics sought in a secret-sharing scheme. That said, the authors seem to be interested in achieving the PSS-PR2 goal of Figure 4 with respect to the adversary class we called $\mathcal{A}_{m,n,t}$.

TOMPA AND WOLL (1986). These authors are interested in $m$-out-of-$n$ threshold schemes that are secure against computationally-unbounded adversaries that can arbitrarily replace the shares of the $m-1$ corrupted players and where some uncorrupted protocol participant is the entity that is recovering the secret [50]. The envisaged adversary is static. The authors state the problem like this (changing only some variable names): *Divide a secret $S \in \{0, 1, \ldots, s-1\}$ into "shares" $S_1, S_2, \ldots, S_n$ such that: (a) Knowledge of any $m$ shares is sufficient to reconstruct $S$ efficiently. (b) Knowledge of $m-1$ shares provides no more information about the value of $S$ that was known before. (c) There is only a small probability $\epsilon > 0$ that any $m-1$ participants $i_1, i_2, \ldots, i_{m-1}$ can fabricate new shares $S'_{i_1}, S'_{i_2}, \ldots, S'_{i_{m-1}}$ that deceive a $m^{\text{th}}$ participant $i_m$. Here, deceiving the $m^{\text{th}}$ participant means that, from $S'_{i_1}, S'_{i_2}, \ldots, S'_{i_{m-1}}$, and $S_{i_m}$, the secret $S'$ reconstructed is "legal" (i.e., $S' \in \{0, 1, \ldots, s-1\}$), but "incorrect" (i.e., $S' \neq S$).* This model is investigated in works like [16, 38], which also addresses some informalities in the definition above (like if the underlying secret $S$ is uniform or if one is instead maximizing over all $S$).

The above goal is approximately translated into our definition for PSS-SR1 (and also demanding perfect-recoverability for erasure adversaries). Note that in a setting like this, with concrete security and a statistical error bound, the difference between static and dynamic adversaries *will* be relevant: one could easily construct an (artificial) secret-sharing scheme with a larger smallest-possible robustness parameter $\epsilon$ if one quantifies over the class of static adversaries instead of dynamic ones.

KRAWCZYK (1993) AND OTHERS. A definition for CSS, for the case of an $n$-out-of-$m$ threshold scheme, was sketched by Krawczyk [31]. It is stated like this, apart from minor changes in notation. *Let $\Pi$ be an $n$-party secret-sharing scheme. For any secret $S$ and for any set of indices $1 \leq i_i \leq \cdots \leq i_r \leq n$ let $\mathcal{D}_\Pi(S, i_1, \ldots, i_r)$ denote the probability distribution on the sequence of shares $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$ induced by the output of running the* Share *algorithm on $S$. The requirement is that for any pair of equal-length secrets $S'$ and $S''$ and any set of indices $i_1, i_2, \ldots, i_r$ with $r < m$, the distributions $\mathcal{D}_\Pi(S', i_1, i_2, \ldots, i_r)$ and $\mathcal{D}_\Pi(S'', i_1, i_2, \ldots, i_r)$ must be polynomially indistinguishable.* Krawczyk earlier indicates that indistinguishability is in terms of *the lengths of messages or secrets*. In Krawczyk's definitional sketch, he omits mention of recoverability. Parameterizing security by in the length of the secret might be unfortunate, effectively excluding a treatment of protocols that share a one-bit secret, say, an apparently legitimate thing to want to do.

A somewhat different approach to formalizing CSS is given by Cachin [14] and refined by Vinod et al. [51]. For privacy one requires that the probability that an adversary can guess the shared secret is negligible (in the security parameterized, which is again the length of the secret). One effectively assumes that the set of secrets is large and that secrets are chosen uniformly from that set (assumptions that seem undesirable). Regardless, an inability to guess the shared secret, an idea going back to Blakley [9], seems to make for an overly weak notion of security, as a huge amount of partial information about the secret might be leaked while the secret remains hard-to-guess. Such considerations are well-known from the context of encryption-scheme privacy, going back to Goldwasser and Micali [22], and they are just as relevant here.

As for the RCSS goal, Krawczyk says only that this is *a secret-sharing scheme that can correctly recover the secret even in the presence of a (bounded) number of corrupted shares, while keeping the secrecy requirement* [31]. Comments in the paper make it clear that the author was thinking in terms of the model of robustness, where an external party recovers the secret.

Krawczyk clearly had further ideas along the lines of those pursued in the current paper. In particular, he indicates that *a stronger definition can be stated in terms of a dynamic and adaptive adversary that progressively chooses the $m-1$ shares to be revealed to him depending on previously opened shares.* He also indicates that *the*

PROCEDURE *Initialize* GSe, GSh
$S \xleftarrow{\$} \mathbb{S}$; $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(S)$

---

PROCEDURE $\mathit{Corrupt}(i)$ GSe, GSh, GSh$_+$, $G$
$T \leftarrow T \cup \{i\}$
RETURN $\boldsymbol{S}[i]$

---

PROCEDURE $\mathit{Finalize}(Y)$ GSe
RETURN $(Y = S)$ AND $T \notin \mathcal{A}$

---

PROCEDURE *Initialize* GSh$_+$
$S \xleftarrow{\$} \mathbb{S}$; $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(S)$
RETURN $S$

---

PROCEDURE *Initialize* $G$
$S^0, S^1 \xleftarrow{\$} \mathbb{S}$; $\boldsymbol{S} \xleftarrow{\$} \mathit{Share}^{\mathit{PSS}}(S^1)$
RETURN $S^0$

---

PROCEDURE $\mathit{Finalize}(j, Y)$ GSh
RETURN $(\boldsymbol{S}[j] = Y)$ AND $(j \notin T)$ AND $T \notin \mathcal{A}$

---

PROCEDURE $\mathit{Finalize}(j, Y)$ GSh$_+$, $G$
RETURN $(\boldsymbol{S}[j] = Y)$ AND $(j \notin T)$ AND $T \cup \{j\} \notin \mathcal{A}$

Figure 9: Games in the PSS lemmas. The Figure defines four games, GSe, GSh, GSh$_+$, and an auxiliary game $G$ to be used in the proofs.

*traditional notion of perfect secret sharing can be defined in an analogous way . . . by replacing "polynomially indistinguishable" with "identical" (or equivalently, by replacing polynomial-time distinguishability tests with computationally unlimited tests)* [31].

# C Secret-Sharing Lemmas

## C.1 Share-prediction lemmas

Assume that a secret is uniformly chosen from a finite set of possible secrets. We consider the probability that an adversary, without having corrupted an authorized subset of players, predicts either the secret that was distributed or the share of an uncorrupted player. The probability of the first is easily shown to be low by the privacy of the scheme, essentially confirming that our definition implies previous ones. Share prediction is more subtle since whether or not it is hard depends on the access structure. We provide sufficient conditions on the access structure for share prediction to have low probability. We give two lemmas, one for adversaries that don't know the secret and one for adversaries that do. The latter is used in our proof of privacy of the HK1 construction (Theorem 1). We consider dynamic adversaries throughout, and in that sense our statements are stronger than in traditional treatments of secret sharing.

We formalize the claims via the games of Figure 9. The Figure shows different procedures, listing next to each the games in which this procedure appears, so that a total of four games are described. For our first lemma, we consider the game GSe whose *Initialize* procedure picks a random secret from the (finite) message space $\mathbb{S}$ of the given PSS scheme $\Pi^{\mathit{PSS}}$ and creates shares for it. The game answers $\mathit{Corrupt}$ queries and declares the adversary to have won if its output $Y$ equals the secret but the set of corrupted players is not authorized. The following says that the probability that the adversary wins is at most $1/|\mathbb{S}|$.

**Lemma 7** Let $\Pi^{\mathit{PSS}} = (\mathit{Share}^{\mathit{PSS}}, \mathit{Recover}^{\mathit{PSS}})$ be a $n$-party PSS scheme over message space $\mathbb{S}$ and access structure $\mathcal{A}$. Then for any adversary $D$

$$\Pr\left[\mathrm{GSe}^D\right] \;\leq\; \frac{1}{|\mathbb{S}|}\,. \tag{15}$$

**Proof:** [Lemma 7]We will specify an adversary $P$ attacking the privacy of $\Pi^{\mathit{PSS}}$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi^{\mathit{PSS}}}(P) \;\geq\; \Pr\left[\mathrm{GSe}^D\right] - \frac{1}{|\mathbb{S}|}\,. \tag{16}$$

Since the advantage of $P$ is 0 by the assumed privacy of the PSS scheme, equation (16) implies equation (15). Adversary $P$ picks $S^0, S^1$ at random from $\mathbb{S}$ and queries $S^0, S^1$ to its *Deal* oracle. It then starts running $A$. When $A$ makes a $Corrupt(i)$ query, adversary $P$ itself makes a $Corrupt(i)$ query, and returns the response to $D$. When $D$ halts with output $Y$, adversary $P$ returns 1 if $Y = S^1$ and 0 otherwise. Denoting the output of $P$ by $d$ and the challenge bit chosen by game Priv by $b$ we have

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi^{PSS}}(P) \quad = \quad \Pr\left[\, d = 1 \mid b = 1 \,\right] - \Pr\left[\, d = 1 \mid b = 0 \,\right] \; .$$

Now we claim

$$\Pr\left[\, d = 1 \mid b = 1 \,\right] \quad = \quad \Pr\left[\mathrm{GSe}^D\right] \tag{17}$$

$$\Pr\left[\, d = 1 \mid b = 0 \,\right] \quad \leq \quad \frac{1}{|\mathbb{S}|} \; , \tag{18}$$

from which (16) follows. Equality (17) is evident from the definitions. In the case $b = 0$, adversary $P$ has no information about $S^1$ which is chosen at random from $\mathbb{S}$ and hence the probability that $Y = S^1$ is at most $1/|\mathbb{S}|$, justifying (18). ▌

Our next lemma considers the game GSh whose *Initialize* procedure picks a random secret from the (finite) message space $\mathbb{S}$ of the given PSS scheme $\Pi^{PSS}$ and creates shares for it. The game answers $Corrupt$ queries and declares the adversary to have won if it outputs $j, Y$ such that $Y$ equals the $j$-th share of the secret but no $Corrupt(j)$ query was made. We are interested in bounding the probability that the adversary wins.

However, this probability is not always small. It depends on the access structure. Consider for example the access structure $\mathcal{A}$ that contains just the sets $[n-1]$ and $[n]$ and let $\mathbb{S} = \{0,1\}^k$. Let algorithm $\mathit{Share}^{PSS}(S)$ return $\boldsymbol{S}$ where $\boldsymbol{S}[1], \ldots, \boldsymbol{S}[n-1]$ are chosen at random from $\mathbb{S}$ subject to $\boldsymbol{S}[1] \oplus \cdots \oplus \boldsymbol{S}[n-1] = S$ and $\boldsymbol{S}[n] = 0^k$. Then an adversary that outputs $n, 0^k$ wins with probability 1.

This type of anomaly seems however absent for "natural" access structures, and in particular for the threshold one $\mathcal{A}_{m,n}$. To be general, we define a property of access structures that is sufficient to ensure that the probability of the adversary winning the GSh game is small. We say that $\mathcal{A}$ is *extendible* if for every $T \subseteq [n]$ such that $T \notin \mathcal{A}$, and every $j \notin T$, there exists a $T' \subseteq [n]$ such that $T \cup T' \notin \mathcal{A}$ but $T \cup T' \cup \{j\} \in \mathcal{A}$. That is, $T$ can be extended to an unauthorized subset such that addition of $j$ makes it authorized. We call $T$ an *extension* of $T, j$.

Note that the $\mathcal{A}$ of our example above is not extendible. Indeed if we set $j = n$ and $T = \emptyset$ then $T, j$ has no extension. However, $\mathcal{A}_{m,n}$ is extendible, as are many other natural access structures. The following says that the probability of winning GSh is at most $1/|\mathbb{S}|$ if the access structure is extendible. The interesting aspect of the proof is that it relies on the recoverability of the PSS scheme, not just its privacy. Below, if $\boldsymbol{Y}$ is a share vector then $\mathrm{Opened}(\boldsymbol{Y})$ denotes the set $\{\, i \,:\, \boldsymbol{Y}[i] \neq \Diamond \,\}$ of all indices at which $\boldsymbol{Y}$ is defined.

**Lemma 8** Let $\Pi^{PSS} = (\mathit{Share}^{PSS}, \mathit{Recover}^{PSS})$ be a $n$-party PSS scheme over message space $\mathbb{S}$ and extendible access structure $\mathcal{A}$. Then for any adversary $E$

$$\Pr\left[\mathrm{GSh}^E\right] \;\leq\; \frac{1}{|\mathbb{S}|} \; . \tag{19}$$

**Proof:** [Lemma 7]Consider the following adversary $D$ for the GSe game. It initializes $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$, and then runs $E$. When $E$ makes a $Corrupt(i)$ query, so does $D$. It stores the response as $\boldsymbol{Y}[i]$ and also returns this response to $E$. Eventually, adversary $E$ halts with output $j, Y$. We say this output is *valid* if $\mathrm{Opened}(\boldsymbol{Y}) \notin \mathcal{A}$ and $j \notin \mathrm{Opened}(\boldsymbol{Y})$. If the output is not valid then $D$ returns something arbitrary like $0, \varepsilon$. Else, it lets $\boldsymbol{Y}[j] \leftarrow Y$ and lets $T'$ be an extension of $T, j$, which we know exists by the extendibility assumption on $\mathcal{A}$. For each $i \in T'$ it makes a $Corrupt(i)$ query and stores the response in $\boldsymbol{Y}[i]$. The extendibility property now guarantees that $\mathrm{Opened}(\boldsymbol{Y}) \in \mathcal{A}$, so $D$ runs $\textit{Recover}^{\textit{PSS}}(\boldsymbol{Y})$ to get back a secret $S'$, outputs $S'$, and halts. The extendibility property also guarantees that $T \cup T' \notin \mathcal{A}$ so that $D$ has not corrupted an authorized subset in the case the output of $E$ is valid. Now if the output $j, Y$ of $E$ is valid and satisfies $\boldsymbol{S}[j] = Y$ then $S' = S$. If the output of $E$ is not valid then $E$ does not win. This means that

$$\Pr\left[\mathrm{GSh}^{E}\right] \;\leq\; \Pr\left[\mathrm{GSe}^{D}\right] \;, \tag{20}$$

whence (19) follows from Lemma 7. ▌

An adversary in the $\mathrm{GSh}_{+}$ game has the same share-prediction objective as an adversary in the GSh game but differs in that it gets the secret as input. (The secret is the output of the *Initialize* procedure which by definition becomes the input to the adversary.) Thus we are now asking how hard it is to predict a share when you know the secret. The following lemma bounds the probability that the adversary wins under the same conditions as in Lemma 8. The crucial difference is that in the $\mathrm{GSh}_{+}$ game, the adversary wins only if not just $T$ but $T \cup \{j\}$ is not authorized. In the case $\mathcal{A} = \mathcal{A}_{m,n}$, this means that we allow it to corrupt only $m-2$ players, not $m-1$ as in Lemma 8. Intuitively, this says that giving the adversary the secret is like giving it one extra share from the point of view of its ability to predict other shares.

**Lemma 9** Let $\Pi^{\textit{PSS}} = (\textit{Share}^{\textit{PSS}}, \textit{Recover}^{\textit{PSS}})$ be a $n$-party PSS scheme over message space $\mathbb{S}$ and extendible access structure $\mathcal{A}$. Then for any adversary $F$

$$\Pr\left[\mathrm{GSh}_{+}^{F}\right] \;\leq\; \frac{1}{|\mathbb{S}|} \;. \tag{21}$$

**Proof:** [Lemma 9]We first claim that

$$\Pr\left[\mathrm{GSh}_{+}^{F}\right] \;=\; \Pr\left[G^{F}\right] \;, \tag{22}$$

where game $G$ is defined via Figure 9. Intuitively, this says that providing $F$ the shared secret as input does not help it; it does equally well with a random, independent secret as input. To justify (22) we provide an adversary $P$ attacking the privacy of $\Pi^{\textit{PSS}}$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi^{\textit{PSS}}}(P) \;=\; \Pr\left[\mathrm{GSh}_{+}^{F}\right] - \Pr\left[G^{F}\right] \;. \tag{23}$$

Since the advantage of $P$ is 0 by the assumed privacy of $\Pi^{\textit{PSS}}$, (23) implies (22). Adversary $P$ picks $S^0, S^1$ at random from $\mathbb{S}$ and queries $S^0, S^1$ to its *Deal* oracle. It initializes set $T$ to empty and starts running $F$ on input $S^1$. When $A$ makes a $Corrupt(i)$ query, $P$ puts $i$ in $T$, itself makes a $Corrupt(i)$ query, and returns the response to $F$. When $F$ halts with output $(j, Y)$, adversary $P$ makes a $Corrupt(j)$ query to obtain $\boldsymbol{S}[j]$. If $\boldsymbol{S}[j] = Y$ and $j \notin T$ then $P$ returns 1, else 0. Equation (23) follows because

$$\Pr\left[\, d = 1 \;|\; b = 1 \,\right] = \Pr\left[\mathrm{GSh}_{+}^{F}\right] \qquad \text{and} \qquad \Pr\left[\, d = 1 \;|\; b = 0 \,\right] = \Pr\left[G^{F}\right] \;,$$

where $d$ denotes the output bit of $P$ and $b$ the challenge bit chosen by game Priv.

Note that the set of players corrupted by $P$ is $T \cup \{j\}$ where $T$ is the set of players corrupted by $F$. But if $T \cup \{j\}$ is not authorized, as is required for $F$ to win, then $P$ has not corrupted an authorized player, as is required for it to win. This is where we use the assumption that $F$ wins only if not just $T$ but $T \cup \{j\}$ is not authorized.

To complete the proof we specify an adversary $E$ for game GSh such that

$$\Pr\left[G^{F}\right] \;\leq\; \Pr\left[\mathrm{GSh}^{E}\right] \;.$$

Now (21) follows from Lemma 8. Adversary $E$ picks $S'$ at random from $\mathbb{S}$ and runs $F$ on input $S'$. It answers $F$'s $Corrupt$ queries via its own $Corrupt$ oracle. When $F$ halts with output $j, Y$, adversary $E$ also outputs $j, Y$ and halts. ▌

## C.2 A recoverability lemma

The following result lets one think of perfect recoverability in a more conventional, adversary-free way.

**Lemma 10 [adversary-free recoverability]** Let $\Pi = (\mathit{Share}, \mathit{Recover})$ be a secret-sharing scheme over message space $\mathbb{S}$ that achieves perfect recoverability over the valid access structure $\mathcal{A}$. Suppose $(S, \boldsymbol{S}, T, \boldsymbol{S}', j)$ is $\mathcal{A}$-generable and $\boldsymbol{S}' \geq \boldsymbol{S}''$. Then $\mathit{Recover}(\boldsymbol{S}_{\overline{T}} \sqcup \boldsymbol{S}''_T, j) = S$. ∎

**Proof:** By the validity of $\mathcal{A}$ there is an adversary $A_{S,T,\boldsymbol{S}',j,\boldsymbol{S}''} \in \mathcal{A}$ that calls $\mathit{Deal}(S)$, calls $Corrupt(i)$ for each $i \in T$, then outputs $(\boldsymbol{S}'', j)$. Now $A_{S,T,\boldsymbol{S}',j,\boldsymbol{S}''}$ will win the Rec game iff *Recover* outputs an $S^* \neq S$. But $A_{S,T,\boldsymbol{S}',j,\boldsymbol{S}''}$ never wins the Rec game because $\mathbf{Adv}^{\rm rec}_{\Pi}(A_{S,T,\boldsymbol{S}',j,\boldsymbol{S}''}) = 0$. It follows that $\mathit{Recover}(\boldsymbol{S}'_{\overline{T}} \sqcup \boldsymbol{S}''_T, j) = \mathit{Recover}(\boldsymbol{S}_{\overline{T}} \sqcup \boldsymbol{S}''_T, j) = S$. ∎

# D Proof of Privacy of HK1 (Theorem 1)

We will actually show something stronger than what is claimed in the theorem statement, namely, that the scheme works for any *extendible access structure*, as defined in Appendix C. We will also use the lemmas of that appendix.

**Proof:** [Theorem 1]The proof will use code-based game-playing [4]. A game in this case will consist of an *Initialize* procedure, procedures to respond to adversary oracle queries of *Deal*, $Corrupt$, and *Hash*, and a *Finalize* procedure.

As is usually the case with game-playing proofs, the different games used have many procedures in common. To compact the game descriptions, we accordingly do not describe each game in full but rather describe all procedures used individually, putting next to their name the games in which they appear. Boxed code in a procedure appears in the game if and only if the game name has a box around it. In this way, Figures 10 and 11 describe a total of 10 games, $G_0$–$G_9$. As an example of how to read the figures, the upper left *Initialize* of Figure 10 occurs in games $G_0, G_1, G_2, G_3, G_4, G_6, G_7, G_8$ while the upper right *Initialize* of the same Figure occurs in the remaining two games, namely $G_5, G_9$. The $Corrupt$ and *Finalize* procedures are the same for all games.

We will be building adversaries that will run $A$ as a subroutine, themselves responding to the latter's oracle queries. Game $G_0$ moves us towards this perspective. (Game $G_0$ is specified by the procedures in the left column of Figure 10, with the boxed statement included in the *Deal* procedure.) Our claim is that

$$\mathbf{Adv}^{\rm priv}_{\Pi}(A) \;=\; 2 \cdot \Pr\left[G_0^A\right] - 1 \;.$$

To justify this let us explain what the game does. Its *Initialize* procedure picks the key $K$ and generates shares for it just like in the game defining the privacy of $\Pi$. While, ideally, we would like to pick the response to $\mathit{Hash}(x)$ at the time $x$ is queried to *Hash*, the game picks the values $\mathit{Hash}(\boldsymbol{K}[i]\,\boldsymbol{C}[i])$ up-front in the *Deal* procedure. (This value is represented by $\boldsymbol{H}[i]$. The IF statement in procedure *Deal* ensures consistency, meaning that $\mathit{Hash}(\boldsymbol{K}[i]\,\boldsymbol{C}[i]) = \mathit{Hash}(\boldsymbol{K}[j]\,\boldsymbol{C}[j])$ in case the arguments to *Hash* are the same in both cases.) It does this because it may soon need to provide $\boldsymbol{X}[i]$ as a response to a $Corrupt(i)$ query, and this share depends on $\mathit{Hash}(\boldsymbol{K}[j]\,\boldsymbol{C}[j])$ for *all* $1 \leq j \leq n$. The assignment of $\boldsymbol{H}[i]$ to $\mathit{Hash}(\boldsymbol{K}[i]\,\boldsymbol{C}[i])$ is done only at the time the adversary makes hash oracle query $\boldsymbol{K}[i]\,\boldsymbol{C}[i]$, necessitating the IF statement in the corresponding procedure.

With the goal now being to upper bound $\Pr[G_0^A]$, let us try to provide some intuition for what follows. Suppose we aim to construct an adversary $B$ attacking the privacy of $\Pi^{\mathit{Enc}}$ with advantage at least $\Pr[G_0^A]$. It would run $A$ to get $X^0, X^1$ and pass these to its left-or-right encryption oracle, getting back a ciphertext $C$ encrypting $X^c$, where $c$ was the random challenge bit underlying its privacy game. It could now use $C$ to construct $\boldsymbol{C}$ and then continue to run $A$, answering its oracle queries as $G_0$ does, and then $A$'s prediction of whether it is seeing $X^0$ or $X^1$ would reveal $c$ to $B$. However, adversary $B$ can't answer $A$'s oracle queries because they depend on

PROCEDURE *Initialize* $\quad G_0$–$G_4$, $G_6$–$G_8$

$K \xleftarrow{\$} \{0,1\}^k$ ; $b \xleftarrow{\$} \{0,1\}$

$\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{\textit{PSS}}(K)$

FOR $i \leftarrow 1$ TO $n$ DO $\boldsymbol{Y}[i] \leftarrow \Diamond$

---

PROCEDURE *Deal*$(X^0, X^1)$ $\quad \boxed{G_0}, G_1$

$C \xleftarrow{\$} \textit{Encrypt}_K(X^b)$

$\boldsymbol{C} \xleftarrow{\$} \textit{Share}^{\textit{IDA}}(C)$

FOR $i \leftarrow 1$ TO $n$ DO

  $\boldsymbol{H}[i] \xleftarrow{\$} \{0,1\}^h$

  IF $\exists\, j < i$ : $(\boldsymbol{K}[i]\, \boldsymbol{C}[i] = \boldsymbol{K}[j]\, \boldsymbol{C}[j])$ THEN

    *bad* $\leftarrow$ `true` ; $\boxed{\boldsymbol{H}[i] \leftarrow \boldsymbol{H}[j]}$

  $\boldsymbol{S}_i \xleftarrow{\$} \textit{Share}^{\textit{ECC}}(\boldsymbol{H}[i])$

---

PROCEDURE $\textit{Corrupt}(i)$ $\quad G_0$–$G_9$

$\boldsymbol{Y}[i] \leftarrow \boldsymbol{K}[i]$

$\boldsymbol{X}[i] \leftarrow \boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i] \cdots \boldsymbol{S}_n[i]$

RETURN $\boldsymbol{X}[i]$

---

PROCEDURE *Hash*$(x)$ $\quad G_0, G_1$

*Hash*$[x] \xleftarrow{\$} \{0,1\}^h$

FOR $i \leftarrow 1$ TO $n$ DO

  IF $(x = \boldsymbol{K}[i]\, \boldsymbol{C}[i])$ THEN *Hash*$[x] \leftarrow \boldsymbol{H}[i]$

RETURN *Hash*$[x]$

---

PROCEDURE *Finalize*$(d)$ $\quad G_0$–$G_9$

RETURN $(d = b)$

---

PROCEDURE *Initialize* $\quad G_5, G_9$

$K, K' \xleftarrow{\$} \{0,1\}^k$ ; $b \xleftarrow{\$} \{0,1\}$

$\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{\textit{PSS}}(K')$

FOR $i \leftarrow 1$ TO $n$ DO $\boldsymbol{Y}[i] \leftarrow \Diamond$

---

PROCEDURE *Deal*$(X^0, X^1)$ $\quad G_2$–$G_9$

$C \xleftarrow{\$} \textit{Encrypt}_K(X^b)$

$\boldsymbol{C} \xleftarrow{\$} \textit{Share}^{\textit{IDA}}(C)$

FOR $i \leftarrow 1$ TO $n$ DO

  $\boldsymbol{H}[i] \xleftarrow{\$} \{0,1\}^h$ ; $\boldsymbol{S}_i \xleftarrow{\$} \textit{Share}^{\textit{ECC}}(\boldsymbol{H}[i])$

---

PROCEDURE *Hash*$(x)$ $\quad \boxed{G_2}, G_3$

*Hash*$[x] \xleftarrow{\$} \{0,1\}^h$

FOR $i \leftarrow 1$ TO $n$ DO

  IF $\boldsymbol{Y}[i] \neq \Diamond$ THEN

    IF $(x = \boldsymbol{K}[i]\, \boldsymbol{C}[i])$ THEN *Hash*$[x] \leftarrow \boldsymbol{H}[i]$

  ELSE IF $(x = \boldsymbol{K}[i]\, \boldsymbol{C}[i])$ THEN

    *bad* $\leftarrow$ `true` ; $\boxed{\textit{Hash}[x] \leftarrow \boldsymbol{H}[i]}$

RETURN *Hash*$[x]$

---

PROCEDURE *Hash*$(x)$ $\quad G_4, G_5$

*Hash*$[x] \xleftarrow{\$} \{0,1\}^h$

FOR $i \leftarrow 1$ TO$n$ DO

  IF $\boldsymbol{Y}[i] \neq \Diamond$ THEN

    IF $(x = \boldsymbol{K}[i]\, \boldsymbol{C}[i])$ THEN *Hash*$[x] \leftarrow \boldsymbol{H}[i]$

RETURN *Hash*$[x]$

Figure 10: Procedures for games in the RO-based instantiation of the HK1 scheme, Theorem 1.

---

shares of $K$ and $B$ does not have access to $K$, which is chosen by its privacy game. The obvious way to get around this is to have $B$ pick some new, random $K'$, generate $\boldsymbol{K}$ via *Share*$^{\textit{PSS}}$, and use these, arguing that $A$ will not know the difference due to the privacy of the PSS scheme. But the *Deal* procedure, which we are suggesting $B$ run, needs to know *all* the values $\boldsymbol{K}[1], \ldots, \boldsymbol{K}[n]$ to perform the test in the IF statement. Similarly, the procedure for replying to *Hash* queries needs to test whether a query contains $\boldsymbol{K}[i]$ for some $i$ and thus needs to know all the values $\boldsymbol{K}$ too. But the PSS scheme does not provide privacy if all shares are revealed.

So our goal to implement the above idea is to put the game in a form where responding to $A$'s queries is possible without knowing the shares of any authorized subset of players. (For concreteness, consider the case where the access structure is $\mathcal{A} = \mathcal{A}_{m,n}$. In this case, we want to be able to respond to $A$'s queries knowing only $m-1$ or less shares of $K$.) We do this in a few steps. Games $G_0, G_1$ differ only in statements following the setting of the flag *bad*, meaning are identical-until-*bad* in the terminology of [4], and so by the Fundamental Lemma of Game Playing from that paper we have

$$\begin{aligned}\Pr\left[G_0^A\right] &= \Pr\left[G_1^A\right] + (\Pr\left[G_0^A\right] - \Pr\left[G_1^A\right]) \\ &\leq \Pr\left[G_1^A\right] + \Pr\left[G_1^A \text{ sets } \textit{bad}\,\right] .\end{aligned}$$

```
PROCEDURE Hash(x)                                          G_6
Hash[x] <-$ {0,1}^h
FOR i <- 1 TO n DO
   IF Y[i] ≠ ◊ THEN
      IF (x = K[i] C[i]) THEN Hash[x] <- H[i]
   ELSE IF (x = K[i] C[i]) AND Opened(Y) ∪ {i} ∉ A THEN
      bad <- true
RETURN Hash[x]
```

```
PROCEDURE Hash(x)                                          G_7
Hash[x] <-$ {0,1}^h
FOR i <- 1 TO n DO
   IF Y[i] ≠ ◊ THEN
      IF (x = K[i] C[i]) THEN Hash[x] <- H[i]
   ELSE IF (x = K[i] C[i]) AND Opened(Y) ∪ {i} ∈ A THEN
      bad <- true
RETURN Hash[x]
```

```
PROCEDURE Hash(x)                G_8, G_9
Hash[x] <-$ {0,1}^h
FOR i <- 1 TO n DO
   IF Y[i] ≠ ◊ THEN
      IF (x = K[i] C[i]) THEN
         Hash[x] <- H[i]
   ELSE
      K_i C_i <- x
      Y_x <- Y ; Y_x[i] <- K_i
      L <- Recover^PSS(Y_x)
      IF L = K THEN bad <- true
RETURN Hash[x]
```

Figure 11: More procedures for the games in the proof of Theorem 1. Above, $\mathrm{Opened}(\boldsymbol{Y})$ denotes the set $\{i : \boldsymbol{Y}[i] \neq \Diamond\}$ of all indices at which $\boldsymbol{Y}$ is defined, and by $K_i\, C_i \leftarrow x$ we mean that $x$ is uniquely parsed into its constituents.

---

Consider the experiment in which we pick $K, \boldsymbol{K}$ as in the *Initialize* procedure of $G_1$. For $1 \leq j < i \leq n$ let $E_{j,i}$ denote the event that $\boldsymbol{K}[j] = \boldsymbol{K}[i]$. Consider the adversary $E_{j,i}$ for game GSh that makes a $Corrupt(j)$ query to get $\boldsymbol{K}[j]$, and then outputs $i, \boldsymbol{K}[j]$. Then by Lemma 8 we have

$$\Pr\left[E_{j,i}\right] \;=\; \Pr\left[\mathrm{GSh}^{E_{j,i}}\right] \;\leq\; \frac{1}{2^k}\;.$$

So by the union bound,

$$\Pr\left[G_1^A \text{ sets } \mathit{bad}\,\right] \;\leq\; \Pr\left[\exists\, j < i \,:\, E_{j,i}\right] \;\leq\; \sum_{j<i} \Pr\left[E_{j,i}\right] \;\leq\; \frac{n(n-1)}{2}\frac{1}{2^k}\;.$$

Since the outcome of $G_1$ is not affected by whether or not *bad* is set, this means that the problematic IF statement of the *Deal* procedure can be removed at the cost of a small loss. The *Deal* procedure of $G_2$ makes this change. With the goal of making responses to *Hash* queries possible without having shares of an authorized subset of players, we split the IF statement of the corresponding procedure of $G_1$ into two parts in $G_2$. Now we have

$$\begin{aligned} \Pr\left[G_1^A\right] &= \Pr\left[G_2^A\right] && (24)\\ &= \Pr\left[G_3^A\right] + (\Pr\left[G_2^A\right] - \Pr\left[G_3^A\right]) \\ &\leq \Pr\left[G_3^A\right] + \Pr\left[G_3^A \text{ sets } \mathit{bad}\,\right]\;, && (25)\end{aligned}$$

the last step again by the Fundamental Lemma of Game Playing. The setting of the flag *bad* by the *Hash* procedure of $G_3$ does not affect the game outcome and so we have

$$\Pr\left[G_3^A\right] \;=\; \Pr\left[G_4^A\right]\;.$$

Now notice that $G_4$ does not make reference to unopened shares of $K$. So at this point we claim that the privacy of the PSS scheme implies

$$\Pr\left[G_4^A\right] \;=\; \Pr\left[G_5^A\right]\;, \qquad (26)$$

where $G_5$ differs from $G_4$ only in the *Initialize* procedure which now produces $\boldsymbol{K}$ by sharing not $K$ but an independently and randomly chosen key $K'$.

Let us now justify (26). To do this we build an adversary $P_1$ attacking the privacy of $\Pi^{PSS}$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi^{PSS}}(P_1) \;=\; \Pr\left[G_4^A\right] - \Pr\left[G_5^A\right] \;. \tag{27}$$

But the privacy of $\Pi^{PSS}$ tells us that the advantage of $P_1$ is zero, yielding (26). Adversary $P_1$ begins by picking $K$ and $K'$ at random from $\{0,1\}^k$ and $b$ at random from $\{0,1\}$. It creates $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$. It then queries $K', K$ to its *Deal* oracle. We know that the latter creates a share vector $\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{PSS}(L)$ where $L = K'$ if the challenge bit $b'$ of the oracle is 0 and $L = K$ if $b' = 1$. Now $P_1$ starts running $A$, responding to $A$'s oracle queries as follows. When $A$ makes a *Deal* query $X^0, X^1$, adversary $P_1$ executes the code of the *Deal* procedure of games $G_4, G_5$. When $A$ makes a $Corrupt(i)$ query, $P_1$ itself makes a $Corrupt(i)$ query to obtain share $\boldsymbol{K}[i]$. It then sets $\boldsymbol{X}[i] \leftarrow \boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i]\cdots\boldsymbol{S}_n[i]$ and $\boldsymbol{Y}[i] \leftarrow \boldsymbol{K}[i]$, and returns $\boldsymbol{X}[i]$ to $A$. When $A$ makes a *Hash*$(x)$ query, $P_1$ executes the code of the *Hash* procedure of games $G_4, G_5$ and returns *Hash*$[x]$ to $A$. When $A$ halts and outputs a bit $d$, adversary $P_1$ returns 1 if $b = d$ and 0 otherwise. It is easy to see that (27) is true.

Game $G_5$ uses $C$, an encryption of $X^b$ under $K$, but makes no other reference to $K$. This puts us in the position we wanted above where we can use the privacy of $\Pi^{Enc}$. Namely, we will now specify $B_1$ so that

$$2 \cdot \Pr\left[G_5^A\right] - 1 \;\le\; \mathbf{Adv}^{\mathrm{ind}}_{\Pi^{Enc}}(B_1) \;. \tag{28}$$

Adversary $B_1$ picks $K'$ at random and lets $\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{PSS}(K')$. It creates $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$. It then runs $A$. When $A$ makes a query $X^0, X^1$ to its *Deal* oracle, $B_1$ queries $X^0, X^1$ to its own left-or-right encryption oracle to get back a ciphertext $C \xleftarrow{\$} \mathit{Encrypt}_K(X^b)$, where $b$ is the challenge bit chosen by the left-or-right encryption oracle. Now $B_1$ executes the last three lines of the *Deal* procedure of game $G_5$. When $A$ makes a $Corrupt(i)$ query, $B_1$ can execute the code of the $Corrupt$ procedure of game $G_5$ since it knows $\boldsymbol{K}[i]$. When $A$ makes a *Hash*$(x)$ query, $B_1$ can similarly execute the code of procedure *Hash* of $G_5$ to obtain the reply and return it to $A$. When $A$ halts and outputs a bit $d$, adversary $B_1$ returns $d$. The advantage of $B_1$ is $2\Pr[b = d] - 1$, so (28) is true.

To summarize, at this point we have shown that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \;\le\; \mathbf{Adv}^{\mathrm{ind}}_{\Pi^{Enc}}(B_1) + \frac{n(n-1)}{2^k} + 2 \cdot \Pr\left[G_3^A \text{ sets } \mathit{bad}\,\right] \;. \tag{29}$$

The difficult part of the proof is to bound $\Pr[G_3^A \text{ sets } \mathit{bad}\,]$. For this we use the key-recovery security of $\Pi^{Enc}$.

Let us again first try to give some intuition. The difficulty with applying the privacy of the PSS scheme is that $A$ has information about $C$. Indeed, in the worst case, the ECC could be replication, meaning $\boldsymbol{C}[i] = C$ for all $1 \le i \le n$, so that $A$ would have $C$ after one $Corrupt$ query. If the encryption scheme, like in our one-time-pad example, permitted recovery of the key from a ciphertext, then $A$ could set *bad* in $G_3$ with high probability. For example, suppose the access structure is $\mathcal{A}_{m,n}$ and we are using Shamir's PSS scheme. Adversary $A$ can obtain $m-1$ shares of $K$, then use $K$ and these shares to compute an unopened share $\boldsymbol{K}[i]$, and query $\boldsymbol{K}[i]\,\boldsymbol{C}[i]$ to *Hash*. In this case, however, we could obtain $K$ from this last oracle query and the opened shares by using the recovery procedure of the PSS scheme. But we can't apply this strategy if $A$ sets *bad* after opening only $m-2$ or fewer shares. In that case, however, Lemma 9 applies, saying that even though $A$ knows $K$, it has low probability of predicting an unopened share.

However, in implementing this we face the same difficulties as above. We can't build a key-recovery adversary if it needs to know shares of the challenge key $K$ to simulate $A$. We want instead to use shares of a different, random $K'$. But for this to be justifiable via the security of the PSS scheme, the game must refer only to opened shares, and $G_3$ does not do this. We now proceed to resolve these problems.

We begin by splitting the bad event into two, one for the case where the set of corrupted players together with the player indicated in the query setting *bad* do not form an authorized subset, and the other where they do:

$$\Pr\left[G_3^A \text{ sets } \textit{bad}\right] \;=\; \Pr\left[G_6^A \text{ sets } \textit{bad}\right] + \Pr\left[G_7^A \text{ sets } \textit{bad}\right] \;.$$

To get some intuition, consider again the case where the access structure is $\mathcal{A}_{m,n}$. Then the first case corresponds to *bad* being set with $m-2$ or less shares opened, and the second the case where $m-1$ shares were open.

We claim Lemma 9 implies

$$\Pr\left[G_6^A \text{ sets } \textit{bad}\right] \;\leq\; \frac{q}{2^k} \;. \tag{30}$$

Let us justify this. For each $j$ in the range $1 \leq j \leq q$ we consider the following adversary $F_j$ for the $\mathrm{GSh}_+$ game. It gets as input a key $K$ chosen at random from $\{0,1\}^k$ by the game, and, via a $Corrupt(i)$ query, can obtain $\boldsymbol{K}[i]$, where $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{\textit{PSS}}(K)$ were generated by the $\mathrm{GSh}_+$ game. $F_j$ begins by creating $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$. It then picks a bit $b$ at random, and initializing a counter $c$ to 0. It then runs $A$. When $A$ makes a query $X^0, X_1$ to its *Deal* oracle, $F_j$ executes the code of the *Deal* procedure of game $G_6$, which it can do since it knows $K$. When $A$ makes a query $i$ to its $Corrupt$ oracle, $F_j$ obtains $\boldsymbol{K}[i]$ via a corrupt query and then executes the code of the $Corrupt$ procedure of $G_6$. When $A$ makes a query $x$ to its *Hash* oracle, $F_j$ does the following:

$c \leftarrow c+1\,;\ \textit{Hash}[x] \xleftarrow{\$} \{0,1\}^h$
FOR $i \leftarrow 1$ TO $n$ DO
  IF $\boldsymbol{Y}[i] \neq \Diamond$ THEN
    IF $(x = \boldsymbol{K}[i]\,\boldsymbol{C}[i])$ THEN $\textit{Hash}[x] \leftarrow \boldsymbol{H}[i]$
  ELSE IF $(c = j)$ THEN $K_j\, C_j \leftarrow x$
RETURN $\textit{Hash}[x]$

Above, by $K_j\, C_j \leftarrow x$ we mean that $x$ is uniquely parsed into its constituents. When $A$ has terminated, algorithm $F_j$ returns $K_j$ and halts. Then

$$\Pr\left[G_6^A \text{ sets } \textit{bad}\right] \;\leq\; \sum_{j=1}^{q} \Pr\left[\mathrm{GSh}_+^{F_j}\right] \;\leq\; \sum_{j=1}^{q} \frac{1}{2^k} \;=\; \frac{q}{2^k} \;,$$

yielding (30). Above, the second inequality is by Lemma 9.

If *bad* is set in $G_7$ then $\mathrm{Opened}(\boldsymbol{Y}_x) = \{\, i : \boldsymbol{Y}_x[i] \neq \Diamond \,\}$ is an authorized subset and hence by the recoverability properties of $\Pi^{\textit{PSS}}$, applying $\textit{Recover}^{\textit{PSS}}$ to $\boldsymbol{Y}_x$ is guaranteed to return the secret $K$ in $G_8$. Thus

$$\Pr\left[G_7^A \text{ sets } \textit{bad}\right] \;\leq\; \Pr\left[G_8^A \text{ sets } \textit{bad}\right] \;. \tag{31}$$

Now, once again, we have managed to create a game, namely $G_8$, that does not reference any unopened share, and are thus in a position to apply the privacy of $\Pi^{\textit{PSS}}$, which we claim implies

$$\Pr\left[G_8^A \text{ sets } \textit{bad}\right] \;=\; \Pr\left[G_9^A \text{ sets } \textit{bad}\right] \;. \tag{32}$$

Note $G_9$ differs from $G_8$ only in the *Initialize* procedure which generates $\boldsymbol{K}$ not from $K$ but from an independently chosen $K'$. To justify (32) we can again build an adversary $P_2$ such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi^{\textit{PSS}}}(P_2) \;=\; \Pr\left[G_8^A \text{ sets } \textit{bad}\right] - \Pr\left[G_9^A \text{ sets } \textit{bad}\right] \;, \tag{33}$$

obtaining (32) because the advantage of $P_2$ is 0 due to the assumed privacy of $\Pi^{\textit{PSS}}$. Adversary $P_2$ begins by picking $K$ and $K'$ at random from $\{0,1\}^k$ and $b$ at random from $\{0,1\}$. It creates $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$. It then queries $K', K$ to its *Deal* oracle. The latter creates shares $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{\textit{PSS}}(L)$ where $L = K'$ if the challenge bit $b'$ of the oracle is 0 and $L = K$ if $b' = 1$. Now $P_2$ starts running $A$, responding to $A$'s oracle queries as follows. When $A$ makes a *Deal* query $X^0, X^1$, adversary $P_2$ executes the code of the *Deal* procedure of games $G_8, G_9$. When $A$ makes a $Corrupt(i)$ query, $P_2$ itself makes a $Corrupt(i)$ query to obtain share $\boldsymbol{K}[i]$. It then sets $\boldsymbol{X}[i] \leftarrow \boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i]\cdots\boldsymbol{S}_n[i]$ and $\boldsymbol{Y}[i] \leftarrow \boldsymbol{K}[i]$, and returns $\boldsymbol{X}[i]$ to $A$. When $A$ makes a $\textit{Hash}(x)$ query, $P_2$ executes the code of the *Hash* procedure of games $G_8, G_9$ and returns $\textit{Hash}[x]$ to $A$. When $A$ halts and outputs a bit $d$, adversary $P_2$ ignores $d$ and returns 1 iff *bad* was set when it responded to some *Hash* query. It is easy to see that (33) is true.

We will now specify $B_2$ so that

$$\Pr\left[G_9^A \text{ sets } \textit{bad}\right] \;\leq\; qn \cdot \mathbf{Adv}^{\text{key}}_{\Pi^{\textit{Enc}}}(B_2)\,. \tag{34}$$

Recall that the key-recovery game picks at random a key $K$ and provides $B_2$ with an encryption oracle $\textit{Encrypt}_K(\cdot)$. Adversary $B_2$ picks $K'$ at random and lets $\boldsymbol{K} \xleftarrow{\$} \textit{Share}^{\textit{PSS}}(K')$. It creates $n$-vector $\boldsymbol{Y}$ to have all components $\Diamond$ and picks bit $b$ at random. It initializes a counter $c$ to 0. It then picks a guess $g_1 \xleftarrow{\$} [q]$ and a guess $g_2 \xleftarrow{\$} [n]$. It then runs $A$. When $A$ makes a query $X^0, X^1$ to its *Deal* oracle, adversary $B_2$ queries $X^b$ to its encryption oracle to get back an encryption $C$ of $X^b$ under $K$. Now $B_2$ executes the last three lines of the *Deal* procedure of game $G_9$. When $A$ makes a $Corrupt(i)$ query, adversary $B_2$ can execute the code of the $Corrupt$ procedure of game $G_5$ since it knows $\boldsymbol{K}[i]$. When $A$ makes a $\textit{Hash}(x)$ query, adversary $B_2$ does the following:

$c \leftarrow c+1\,;\ \textit{Hash}[x] \xleftarrow{\$} \{0,1\}^h$
FOR $i \leftarrow 1$ TO $n$ DO
  IF $\boldsymbol{Y}[i] \neq \Diamond$ THEN
    IF $(x = \boldsymbol{K}[i]\,\boldsymbol{C}[i])$ THEN $\textit{Hash}[x] \leftarrow \boldsymbol{H}[i]$
  ELSE IF $(c,i) = (g_1, g_2)$ THEN
    $K_i\,C_i \leftarrow x\,;\ \boldsymbol{Y}_x \leftarrow \boldsymbol{Y}\,;\ \boldsymbol{Y}_x[i] \leftarrow K_i\,;\ L \leftarrow \textit{Recover}^{\textit{PSS}}(\boldsymbol{Y}_x)$
RETURN $\textit{Hash}[x]$

That is, when $(c,i)$ is equal to $(g_1, g_2)$, adversary $B_2$ records the candidate key as $L$. When $A$ has terminated, adversary $B_2$ returns $L$ and halts. One can check that (34) is true.

In summary, this second part of the proof has shown that

$$\Pr\left[G_3^A \text{ sets } \textit{bad}\right] \;\leq\; \frac{q}{2^k} + qn \cdot \mathbf{Adv}^{\text{key}}_{\Pi^{\textit{Enc}}}(B_2)\,.$$

Combining this with (29) completes the proof of the theorem. ▮

# E Proof of Recoverability of HK1 (Theorem 3)

**Proof:** [Theorem 3]Let $\Pi = (\textit{Share}, \textit{Recover})$, $\Pi^{\textit{Enc}} = (\textit{Encrypt}, \textit{Decrypt})$, $\Pi^{\textit{PSS}} = (\textit{Share}^{\textit{PSS}}, \textit{Recover}^{\textit{PSS}})$, $\Pi^{\textit{IDA}} = (\textit{Share}^{\textit{IDA}}, \textit{Recover}^{\textit{IDA}})$, and $\Pi^{\textit{ECC}} = (\textit{Share}^{\textit{ECC}}, \textit{Recover}^{\textit{ECC}})$. Consider running $A$ with game Rec. Let $K, C, \boldsymbol{K}, \boldsymbol{C}, \boldsymbol{H}, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n, \boldsymbol{X}$ denote the quantities chosen by the *Share* algorithm when it is executed by the *Deal* procedure in response to $A$'s *Deal* query of $X$. Let $(\boldsymbol{X}', j)$ denote the output of $A$. Let $K', C', \boldsymbol{K}', \boldsymbol{C}', \boldsymbol{H}', \boldsymbol{S}'_1, \ldots, \boldsymbol{S}'_n, X'$ denote, respectively, the quantities $K, C, \boldsymbol{K}, \boldsymbol{C}, \boldsymbol{H}, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n, X$ as defined by $\textit{Recover}(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T, j)$ when it is executed by the *Finalize* procedure of Rec, where $T$ is the set of players that $A$ corrupted. We consider the following events:

$E_1$: $\exists \ell \in [n]$ such that $\boldsymbol{H}[\ell] \neq \boldsymbol{H}'[\ell]$
$E_2$: $\exists \ell \in T$ such that $\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell] \notin \{\Diamond\,\Diamond, \boldsymbol{K}[\ell]\,\boldsymbol{C}[\ell]\}$
$E_3$: $K \neq K'$
$E_4$: $C \neq C'$

If $C = C'$ and $K = K'$ then the secret $X'$ that is recovered equals $X$ so

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) &\leq \Pr[E_3 \vee E_4] \\
&\leq \Pr[E_1 \vee E_2 \vee E_3 \vee E_4] \\
&= \Pr[E_1] + \Pr[\overline{E}_1 \wedge E_2] + \Pr[\overline{E}_1 \wedge \overline{E}_2 \wedge E_3] + \Pr[\overline{E}_1 \wedge \overline{E}_2 \wedge \overline{E}_3 \wedge E_4] \\
&\leq \Pr[E_1] + \Pr[\overline{E}_1 \wedge E_2] + \Pr[\overline{E}_2 \wedge E_3] + \Pr[\overline{E}_2 \wedge E_4]\ . \qquad (35)
\end{aligned}$$

We bound each addend above in turn. Let $E_{1,\ell}$ be the event that $\boldsymbol{H}[\ell] \neq \boldsymbol{H}'[\ell]$. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{S}'_\ell[i] = \boldsymbol{S}_\ell[i]$ by line 21 in Figure 5. But $\boldsymbol{S}_\ell$ is an output of $\mathit{Share}^{ECC}(\boldsymbol{H}[\ell])$ and $\overline{T} \in \mathcal{A}$, so $\mathit{Recover}^{ECC}(\boldsymbol{S}'_\ell, j) = \boldsymbol{H}[\ell]$ by Lemma 10 applied to $\Pi^{ECC}$, meaning $\boldsymbol{H}'[\ell] = \boldsymbol{H}[\ell]$. So $\Pr[E_{1,\ell}] = 0$. Now by the union bound we have

$$\Pr[E_1] \;\leq\; \sum_{\ell=1}^{n} \Pr[E_{1,\ell}] \;=\; 0\ . \qquad (36)$$

Next we claim that

$$\Pr[E_2] \;\leq\; \frac{(q+2n)^2}{2^{h+1}}\ . \qquad (37)$$

We justify this as follows. Suppose $\ell \in T$ and $\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell] \neq \Diamond\,\Diamond$. By lines 21 and 25 of Figure 5 it must be that $\mathit{Hash}(\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell]) = \boldsymbol{H}[\ell]$. But if $\overline{E}_1$ then $\boldsymbol{H}'[\ell] = \boldsymbol{H}[\ell]$, and by line 14 of Figure 5 we know that $\boldsymbol{H}[\ell] = \mathit{Hash}(\boldsymbol{K}[\ell]\,\boldsymbol{C}[\ell])$. So we have $\mathit{Hash}(\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell]) = \mathit{Hash}(\boldsymbol{K}[\ell]\,\boldsymbol{C}[\ell])$. Thus if $\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell] \neq \boldsymbol{K}[\ell]\,\boldsymbol{C}[\ell]$ then we have a collision in $\mathit{Hash}$. Thus if $\overline{E}_1 \wedge E_2$ we have found a collision in $\mathit{Hash}$. At this point we need only bound the probability of a collision in $\mathit{Hash}$. The random-oracle $\mathit{Hash}$ is invoked at most $q + 2n$ times, justifying (37).

Next we claim that

$$\Pr[\overline{E}_2 \wedge E_3] = 0\ . \qquad (38)$$

We justify this as follows. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{K}'[i] = \boldsymbol{K}[i]$ by line 21 of Figure 5. If $i \in T$ and $\overline{E}_2$ holds then $\boldsymbol{K}'[i] \in \{\Diamond, \boldsymbol{K}[i]\}$. But $\boldsymbol{K}$ is an output of $\mathit{Share}^{PSS}(K)$ and $\overline{T} \in \mathcal{A}$, so $\mathit{Recover}^{PSS}(\boldsymbol{K}', j) = K$ by Lemma 10 applied to $\Pi^{PSS}$, meaning $K' = K$. So $E_3$ cannot hold.

Finally, we claim that

$$\Pr[\overline{E}_2 \wedge E_4] = 0\ . \qquad (39)$$

We justify this as follows. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{C}'[i] = \boldsymbol{C}[i]$ by line 21 of Figure 5. If $i \in T$ and $\overline{E}_2$ holds then $\boldsymbol{C}'[i] \in \{\Diamond, \boldsymbol{C}[i]\}$. But $\boldsymbol{C}$ is an output of $\mathit{Share}^{IDA}(C)$ and $\overline{T} \in \mathcal{A}$, so $\mathit{Recover}^{IDA}(\boldsymbol{C}', j) = C$ by Lemma 10 applied to $\Pi^{IDA}$, meaning $C' = C$. So $E_4$ cannot hold.

Putting together equations (35)–(39) completes the proof. ▌

# F Proof of Recoverability of HK2 (Theorem 5)

**Proof:** [Theorem 5]Let $\Pi = (\mathit{Share}, \mathit{Recover})$, $\Pi^{Enc} = (\mathit{Encrypt}, \mathit{Decrypt})$, $\Pi^{PSS} = (\mathit{Share}^{PSS}, \mathit{Recover}^{PSS})$, $\Pi^{IDA} = (\mathit{Share}^{IDA}, \mathit{Recover}^{IDA})$, and $\Pi^{ECC} = (\mathit{Share}^{ECC}, \mathit{Recover}^{ECC})$. Consider running $A$ with game Rec. Let $K, C, \boldsymbol{K}, \boldsymbol{C}, \boldsymbol{H}, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n, \boldsymbol{X}$ denote the quantities chosen by the $\mathit{Share}$ algorithm when it is executed by the $\mathit{Deal}$ procedure in response to $A$'s $\mathit{Deal}$ query of $X$. Let $(\boldsymbol{X}', j)$ denote the output of $A$. Let

PROCEDURE $Corrupt(i)$
RETURN $\boldsymbol{X}[i]$

PROCEDURE *Finalize* $(\boldsymbol{X}', j)$
FOR $i \leftarrow 1$ TO $n$ DO
  $\boldsymbol{R}'[i]\boldsymbol{K}'[i]\boldsymbol{C}'[i]\ \boldsymbol{S}'_1[i]\boldsymbol{S}'_2[i]\cdots\boldsymbol{S}'_n[i] \leftarrow \boldsymbol{X}'[i]$
RETURN $(\boldsymbol{K}'[\ell]\,\boldsymbol{C}'[\ell], \boldsymbol{R}'[\ell])$

PROCEDURE *Deal*$(X)$
$\ell \xleftarrow{\$} [n]\,;\ K \xleftarrow{\$} \{0,1\}^k\,;\ C \xleftarrow{\$} \mathit{Encrypt}_K(X)$
$\boldsymbol{K} \xleftarrow{\$} \mathit{Share}^{PSS}(K)\,;\ \boldsymbol{C} \xleftarrow{\$} \mathit{Share}^{IDA}(C)$
FOR $i \leftarrow 1$ TO $n$ DO
  IF $i = \ell$ THEN $(\boldsymbol{H}[\ell], \boldsymbol{R}[\ell]) \xleftarrow{\$} \mathit{Commit}(\boldsymbol{K}[\ell]\,\boldsymbol{C}[\ell])$
  ELSE $(\boldsymbol{H}[i], \boldsymbol{R}[i]) \xleftarrow{\$} \mathit{Ct}(\boldsymbol{K}[i]\,\boldsymbol{C}[i])$
  $\boldsymbol{S}_i \xleftarrow{\$} \mathit{Share}^{ECC}(\boldsymbol{H}[i])$
FOR $i \leftarrow 1$ TO $n$ DO
  $\boldsymbol{X}[i] \leftarrow \boldsymbol{R}[i]\boldsymbol{K}[i]\boldsymbol{C}[i]\ \boldsymbol{S}_1[i]\cdots\boldsymbol{S}_n[i]$

Figure 12: Procedures used by adversary $A_{BIND}$ to respond to oracle queries of $A$ in the proof of Theorem 5.

$K', C', \boldsymbol{K}', \boldsymbol{C}', \boldsymbol{H}', \boldsymbol{S}'_1, \ldots, \boldsymbol{S}'_n, X'$ denote, respectively, the quantities $K, C, \boldsymbol{K}, \boldsymbol{C}, \boldsymbol{H}, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n, X$ as defined by *Recover*$(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T, j)$ when it is executed by the *Finalize* procedure of Rec, where $T$ is the set of players that $A$ corrupted. We consider the following events:

$E_1$: $\exists \ell \in [n]$ such that $\boldsymbol{H}[\ell] \neq \boldsymbol{H}'[\ell]$
$E_2$: $\exists \ell \in T$ such that $\boldsymbol{K}'[\ell]\ \boldsymbol{C}'[\ell] \notin \{\Diamond\,\Diamond, \boldsymbol{K}[\ell]\ \boldsymbol{C}[\ell]\}$
$E_3$: $K \neq K'$
$E_4$: $C \neq C'$

If $C = C'$ and $K = K'$ then the secret $X'$ that is recovered equals $X$ so

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rec}}_{\Pi}(A) &\leq \Pr[E_3 \vee E_4] \\
&\leq \Pr[E_1 \vee E_2 \vee E_3 \vee E_4] \\
&= \Pr[E_1] + \Pr[\overline{E}_1 \wedge E_2] + \Pr[\overline{E}_1 \wedge \overline{E}_2 \wedge E_3] + \Pr[\overline{E}_1 \wedge \overline{E}_2 \wedge \overline{E}_3 \wedge E_4] \\
&\leq \Pr[E_1] + \Pr[\overline{E}_1 \wedge E_2] + \Pr[\overline{E}_2 \wedge E_3] + \Pr[\overline{E}_2 \wedge E_4]\,. \qquad (40)
\end{aligned}$$

We bound each addend above in turn. Let $E_{1,\ell}$ be the event that $\boldsymbol{H}[\ell] \neq \boldsymbol{H}'[\ell]$. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{S}'_\ell[i] = \boldsymbol{S}_\ell[i]$ by line 21 in Figure 7. But $\boldsymbol{S}_\ell$ is an output of *Share*$^{ECC}(\boldsymbol{H}[\ell])$ and $\overline{T} \in \mathcal{A}$, so *Recover*$^{ECC}(\boldsymbol{S}'_\ell, j) = \boldsymbol{H}[\ell]$ by Lemma 10 applied to $\Pi^{ECC}$, meaning $\boldsymbol{H}'[\ell] = \boldsymbol{H}[\ell]$. So $\Pr[E_{1,\ell}] = 0$. Now by the union bound we have

$$\Pr[E_1] \;\leq\; \sum_{\ell=1}^{n} \Pr[E_{1,\ell}] \;=\; 0\,. \qquad (41)$$

Next we construct adversary $B$ such that

$$\Pr[\overline{E}_1 \wedge E_2] \;\leq\; n \cdot \mathbf{Adv}^{\mathrm{bind}}_{\Pi^{Com}}(B)\,. \qquad (42)$$

Adversary $B$ runs $A$, responding to its *Deal* and $Corrupt$ oracle calls via the procedures of Figure 12. When $A$ halts with output $(\boldsymbol{X}', j)$, adversary $B$ runs the Finalize procedure of the same figure.

Next we claim that

$$\Pr[\overline{E}_2 \wedge E_3] = 0\,. \qquad (43)$$

We justify this as follows. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{K}'[i] = \boldsymbol{K}[i]$ by line 21 of Figure 7. If $i \in T$ and $\overline{E}_2$ holds then $\boldsymbol{K}'[i] \in \{\Diamond, \boldsymbol{K}[i]\}$. But $\boldsymbol{K}$ is an output of *Share*$^{PSS}(K)$ and $\overline{T} \in \mathcal{A}$, so *Recover*$^{PSS}(\boldsymbol{K}', j) = K$ by Lemma 10 applied to $\Pi^{PSS}$, meaning $K' = K$. So $E_3$ cannot hold.

Finally, we claim that

$$\Pr[\overline{E}_2 \wedge E_4] = 0\,. \qquad (44)$$

| | |
|---|---|
| PROCEDURE $A$<br>Run $B$<br>When $B$ makes a query *Enc*$(M)$<br>$X^1 \leftarrow M\,;\; X^0 \leftarrow \overline{M}$<br>*Deal*$(X^0, X^1)$<br>$\boldsymbol{X}[1] \leftarrow Corrupt(1)$<br>$\boldsymbol{K}[1]\; C\; h_1 h_2 h_3 \leftarrow \boldsymbol{X}[1]$<br>Return $C$ to $B$<br>When $B$ outputs $K'$<br>$\boldsymbol{K}'[2] \leftarrow R(\boldsymbol{K}[1], K')$<br>IF *Hash*$(\boldsymbol{K}'[2]\; C) = h_2$ THEN<br>$X \leftarrow$ *Decrypt*$_{K'}(C)$<br>IF $X = X^1$ THEN RETURN 1 ELSE RETURN 0<br>RETURN 0 | PROCEDURE $A$<br>Run $B$<br>When $B$ makes a query *LeftOrRight*$(X^0, X^1)$<br>*Deal*$(X^0, X^1)$<br>$\boldsymbol{X}[1] \leftarrow Corrupt(1)$<br>$\boldsymbol{K}[1]\; C\; h_1 h_2 h_3 \leftarrow \boldsymbol{X}[1]$<br>Return $C$ to $B$<br>When $B$ outputs $b'$<br>RETURN $b'$ |

Figure 13: Adversaries for establishing the minimality of the ind1+key1 assumption for the privacy of HK1, Theorem 2.

We justify this as follows. If $i \notin T$ then $(\boldsymbol{X}_{\overline{T}} \sqcup \boldsymbol{X}'_T)[i] = \boldsymbol{X}[i]$ and hence $\boldsymbol{C}'[i] = \boldsymbol{C}[i]$ by line 21 of Figure 7. If $i \in T$ and $\overline{E}_2$ holds then $\boldsymbol{C}'[i] \in \{\Diamond, \boldsymbol{C}[i]\}$. But $\boldsymbol{C}$ is an output of *Share*$^{IDA}(C)$ and $\overline{T} \in \mathcal{A}$, so *Recover*$^{IDA}(\boldsymbol{C}', j) = C$ by Lemma 10 applied to $\Pi^{IDA}$, meaning $C' = C$. So $E_4$ cannot hold.

Putting together equations (40)–(44) completes the proof. ▌

## G Proof of Theorem 2

We use the same approach as in our attack on HK1, except that the one-time pad is replaced by the given encryption scheme $\Pi^{Enc}$. So let $n = 3$, $m = 2$, let $\Pi^{PSS}$ be Shamir's scheme over $\mathbb{F}_{2^{128}}$ for $\mathcal{A} = \mathcal{A}_{2,3}$, and let $\Pi^{IDA}$ and $\Pi^{ECC}$ both be replication. Let $R$ denote the algorithm such that $R(\boldsymbol{K}[1], K) = \boldsymbol{K}[2]$ for all $\boldsymbol{K} \in$ *Share*$^{PSS}(K)$. (We already discussed that Shamir's scheme admits an efficient such $R$.)

The adversary $A$ for the proof of part (1) is shown on the left-hand side of Figure 13. We write $\overline{M}$ for the bitwise complement of $M$ (here, an arbitrary string distinct from $M$). We proceed to the analysis. Let PrivL and PrivR denote the games that are the same as the Priv game except the encryption oracle *Deal* is replaced by the oracle that always deals the left or right queries, respectively. Let $K$ denote the underlying key chosen by the *Deal* oracle. Then

$$\begin{aligned}\Pr[\text{PrivR}^A \Rightarrow 1] &\geq \Pr\left[\,\text{PrivR}^A \Rightarrow 1 \mid K = K'\,\right] \cdot \Pr[K = K'] \\ &= 1 \cdot \mathbf{Adv}^{\text{key}}_{\Pi^{Enc}}(B)\end{aligned}$$

and

$$\begin{aligned}&\Pr[\text{PrivL}^A \Rightarrow 1] \\ &\quad= \Pr\left[\,\text{PrivL}^A \Rightarrow 1 \mid K = K'\,\right] \cdot \Pr[K = K'] + \Pr\left[\,\text{PrivL}^A \Rightarrow 1 \mid K \neq K'\,\right] \cdot \Pr[K \neq K'] \\ &\quad= 0 \cdot \Pr[K = K'] + 2^{-h} \cdot \Pr[K \neq K'] \\ &\quad\leq 2^{-h}\,.\end{aligned}$$

Thus

$$\mathbf{Adv}^{\text{priv}}_{\Pi}(A) \;=\; \Pr[\text{PrivR}^A \Rightarrow 1] - \Pr[\text{PrivL}^A \Rightarrow 1] \;\geq\; \mathbf{Adv}^{\text{key}}_{\Pi^{Enc}}(B) - 2^{-h}\,.$$

This completes the proof for part (1).

The adversary $A$ for the proof of part (2) is shown on the right-hand side of Figure 13. The analysis is straightforward and omitted.