Lectures on NIZKs:
A Concrete Security Treatment

Mihir Bellare[1]

August 3, 2021

Abstract

We initiate a concrete-security treatment of NIZKs. We start with definitions conducive to such a treatment. We give some basic general results on, and relations between, these notions. We then look at some NIZK systems in the literature and give concrete (as opposed to asymptotic) results about whatever properties they possess. We explore some basic applications of NIZKs, such as digital signatures, to give concrete security reductions. We define dual-mode proof systems as a way to formalize ideas underlying some NIZKs in the literature. This concrete security treatment of NIZKs is motivated by emerging applications, where it serves to help determine, and also reduce, parameters for a given level of security, leading to security-preserving efficiency gains. These notes were created as part of a course, taught in Winter 2020, on NIZKs. They represent a work in progress intended as a starting point for a future paper, not a finished product.

[1] Department of Computer Science & Engineering, University of California San Diego, USA. Email: `mihir@eng.ucsd.edu`. URL: `cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grant CNS-1228890, NSF grant CNS-1526801, ERC Project ERCC FP7/615074 and a gift from Microsoft corporation.

Contents

1 Introduction

Non-interactive zero-knowledge (NIZK) systems are seeing increasing usage and application, making efficiency a target. But efficiency decoupled from security becomes nonsensical. The proper perspective is that we have some desired level of security (for example, 256 bits), for some set of goals (for example, soundness and zero-knowledge) and then want to minimize cost subject to staying at this level of security. To perform a design task like this in a rigorous way, we need quantitative metrics and results for NIZK security. This work initiates a treatment of NIZKs that provides this.

Zero-knowledge was introduced by [GMR89]. NIZKs were introduced by [BFM88, BDSMP91]. (The latter paper is the definitive reference, fixing the many bugs in the former.) Many constructions [FLS99, BY96, KP98] and applications [BG90, DDN00] followed. The treatment here was asymptotic. Security (for whatever notion) meant that polynomial-time adversaries have a success probability that is a negligible function of the security parameter. Such a treatment does not facilitate determining what setting of parameters provides some desired numeric level of security.

Applications have grown, fueling a search for efficient NIZKs. The Groth-Sahai framework [GS08] is widely utilized. We are seeing both new protocols and their implementation [GS08, Gro10, BCTV14, EG14]. Structure-preserving cryptography [AFG$^+$10, AGOT14, Gro15] was developed to allow these NIZKs to be used for efficient applications.

In these notes, we initiate a concrete security treatment of NIZKs. We return to the basic definitions and cast them via games and adversary advantage functions. We give quantitative results about relations between notions. We then cast some existing proof systems in this framework. And we look at the concrete security of reductions for applications that use NIZKs.

These notes are a work in progress, meant for a class I am teaching at UCSD in Winter 2020. They are in constant flux. Not only may things be added, but things can change. There will be mistakes, big and small.

In CSE207, the treatment was concrete. Students entering with only that background may not know what is an asymptotic treatment. No matter. For you, the current treatment of NIZKs may seem more natural.

There is some novelty in the definitions given here, as discussed further in Section 4. For example, we give joint definitions of ZK + soundness/extractability, allow multiple verification/extraction attempts, and formulate dual-mode NIZKs.

The notes include explicit Exercises. But they also include implicit ones, indicated sometimes just by the question "why?" somewhere in the text, prompting the reader to do the exercise of answering the question.

Comments, corrections, feedback, thoughts and suggestions, whether technical, historical or opinionated, are welcome.

2 Asymptotic, concrete and blended security: An example

This section introduces the reader to the asymptotic, concrete and blended settings for cryptographic definitions and proofs via a simple example. We define the OW (one-wayness) and PR (pseudo-randomness) security of a function, and show that the latter (under some conditions) implies the former.

2.1 Asymptotic

A function $\nu\colon \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p\colon \mathbb{N} \to \mathbb{R}$ there is a $\lambda_p \in \mathbb{N}$ such that $\nu(\lambda) \leq 1/p(\lambda)$ for all $\lambda \geq \lambda_p$. "PT" stands for "polynomial time." By 1^λ we denote the unary representation of the integer security parameter $\lambda \in \mathbb{N}$.

A generator is a PT deterministic algorithm G that takes 1^λ and an input $X \in \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}$ to return an output $\mathsf{G}(1^\lambda, X) \in \{0,1\}^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$. Here $\mathsf{G}.\mathrm{il}, \mathsf{G}.\mathrm{ol}\colon \mathbb{N} \to \mathbb{N}$ are PT deterministic functions determining the input and output lengths. We say that G is OW-secure (OW stands for "One-Way") if for every PT adversary I the function

$$\Pr\Big[\, \mathsf{G}(1^\lambda, X') = Y \;\;:\;\; X \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}\,;\; Y \leftarrow \mathsf{G}(1^\lambda, X)\,;\; X' \leftarrow_{\$} \mathrm{I}(1^\lambda, Y)\,\Big]$$

is negligible. We say that G is PR-secure (PR for "Pseudo-Random") if for every PT adversary D the function

$$\Pr\Big[\, \mathrm{D}(1^\lambda, Y_b) = b \;\;:\;\; b \leftarrow_{\$} \{0,1\}\,;\; X \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}\,;\; Y_1 \leftarrow \mathsf{G}(1^\lambda, X)\,;\; Y_0 \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{ol}(\lambda)}\,\Big] - \frac{1}{2}$$

is negligible.

Theorem 2.1 *Let G be a generator such that the function $\ell(\lambda) = 2^{\mathsf{G}.\mathrm{il}(1^\lambda) - \mathsf{G}.\mathrm{ol}(1^\lambda)}$ is negligible. Assume G is PR-secure. Then G is OW-secure.*

Proof of Theorem 2.1: Suppose towards a contradiction that G is not OW-secure. Then there exists a PT adversary I and a positive polynomial $p_{\mathrm{I}}\colon \mathbb{N} \to \mathbb{R}$ such that

$$\Pr\Big[\, \mathsf{G}(1^\lambda, X') = Y \;\;:\;\; X \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}\,;\; Y \leftarrow \mathsf{G}(1^\lambda, X)\,;\; X' \leftarrow_{\$} \mathrm{I}(1^\lambda, Y)\,\Big] \geq \frac{1}{p_{\mathrm{I}}(\lambda)} \tag{1}$$

for infinitely many $\lambda \in \mathbb{N}$. We show that there exists a PT adversary D and a polynomial $p_{\mathrm{D}}\colon \mathbb{N} \to \mathbb{R}$ such that

$$\Pr\Big[\, \mathrm{D}(1^\lambda, Y_b) = b \;\;:\;\; b \leftarrow_{\$} \{0,1\}\,;\; X \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}\,;\; Y_1 \leftarrow \mathsf{G}(1^\lambda, X)\,;\; Y_0 \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{ol}(\lambda)}\,\Big]$$
$$\geq \frac{1}{2} + \frac{1}{p_{\mathrm{D}}(\lambda)}$$

for infinitely many $\lambda \in \mathbb{N}$. This contradicts the assumption that G is PR-secure, establishing the Theorem.

Adversary $\mathrm{D}(1^\lambda, Y)$ runs $\mathrm{I}(1^\lambda, Y)$ to get X'. If $\mathsf{G}(1^\lambda, X') = Y$ then it outputs 1, else it outputs 0. If $Y = \mathsf{G}(X)$ for $X \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}$ then the probability that D outputs 1 is at least the probability on the left of Equation (1). If $Y \leftarrow_{\$} \{0,1\}^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$ then the probability that D outputs 1 is at most $2^{\mathsf{G}.\mathrm{il}(1^\lambda)}/2^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$ because an X' satisfying $Y = \mathsf{G}(X)$ exists with at most this probability. ▮

2.2 Concrete

A generator is a deterministic algorithm G that takes an input $X \in \{0,1\}^{\mathsf{G}.\mathrm{il}}$ to return an output $\mathsf{G}(X) \in \{0,1\}^{\mathsf{G}.\mathrm{ol}}$. Here $\mathsf{G}.\mathrm{il}, \mathsf{G}.\mathrm{ol} \in \mathbb{N}$ are the input and output lengths. The ow-advantage of an

Game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{G}}$	Game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}$
$\textsc{Init}()$:	$\textsc{Init}()$:
$X \leftarrow\!\!\$\, \{0,1\}^{\mathsf{G.il}}$; $Y \leftarrow \mathsf{G}(X)$	$b \leftarrow\!\!\$\, \{0,1\}$; $X \leftarrow\!\!\$\, \{0,1\}^{\mathsf{G.il}}$; $Y_1 \leftarrow \mathsf{G}(X)$; $Y_0 \leftarrow\!\!\$\, \{0,1\}^{\mathsf{G.ol}}$
Return Y	Return Y_b
$\textsc{Fin}(X')$:	$\textsc{Fin}(b')$:
Return ($\mathsf{G}(X') = Y$)	Return $(b' = b)$

Adversary D:

$Y \leftarrow \mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}.\textsc{Init}$ // Adversary D calls $\textsc{Init}$ of its own game
$\mathrm{I}^{\textsc{Init},\textsc{Fin}}$ // Run adversary I with subroutines below for its oracles

$\textsc{Init}()$: // Subroutine defined by D
Return Y // Reply to I's $\textsc{Init}$ query

$\textsc{Fin}(X')$: // Subroutine defined by D. Here X' is provided by I.
If $\mathsf{G}(X') = Y$ then $b' \leftarrow 1$ else $b' \leftarrow 0$
$\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}.\textsc{Fin}(b')$ // Adversary D calls $\textsc{Fin}$ of its own game

Figure 1: Top: Games defining OW and PR security for generator G in the concrete security setting. Bottom: Adversary D for Theorem 2.2.

---

adversary I is defined by

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G}}(\mathrm{I}) = \Pr[\mathbf{G}^{\mathrm{ow}}_{\mathsf{G}}(\mathrm{I})]$$

where game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{G}}$ is on the top left in Figure 1. The adversary must begin by calling oracle $\textsc{Init}$ (to get Y), and conclude by calling oracle $\textsc{Fin}$ with an argument $X' \in \{0,1\}^{\mathsf{G.il}}$. The notation $\Pr[\mathbf{G}^{\mathrm{ow}}_{\mathsf{G}}(\mathrm{I})]$ refers to the probability that $\textsc{Fin}$ returns `true` in the execution of the game with the adversary. The pr-advantage of an adversary D is defined by

$$\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G}}(\mathrm{D}) = 2\Pr[\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}(\mathrm{D})] - 1$$

where game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}$ is on the top right in Figure 1. The conventions are similar.

Theorem 2.2 *Let G be a generator. Given an adversary I we can construct an adversary D (shown explicitly at the bottom of Figure 1) such that*

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G}}(\mathrm{I}) \le \mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G}}(\mathrm{D}) + \frac{1}{2^{\mathsf{G.ol}-\mathsf{G.il}}} \,. \tag{2}$$

The running time of D is about the same as that of I.

Proof of Theorem 2.2: Let b be the challenge bit chosen at random in game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}$, and b' the bit queried by D to $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}.\textsc{Fin}$. Then (appealing here to Lemma 3.2) we have

$$\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G}}(\mathrm{D}) = \Pr\left[\, b' = 1 \mid b = 1 \,\right] - \Pr\left[\, b' = 1 \mid b = 0 \,\right]$$

where the probabilities are in the execution of game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G}}$ with adversary D. If $b = 1$ and I succeeds,

Game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{G},\lambda}$	Game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}$
$\textsc{Init}()$:	$\textsc{Init}()$:
$X \leftarrow\!\!\$\ \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}$; $Y \leftarrow \mathsf{G}(1^\lambda, X)$	$b \leftarrow\!\!\$\ \{0,1\}$; $X \leftarrow\!\!\$\ \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}$; $Y_1 \leftarrow \mathsf{G}(1^\lambda, X)$; $Y_0 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$
Return $(1^\lambda, Y)$	Return $(1^\lambda, Y_b)$
$\textsc{Fin}(X')$:	$\textsc{Fin}(b')$:
Return $(\ \mathsf{G}(1^\lambda, X') = Y\ )$	Return $(b' = b)$

Adversary D:

$(1^\lambda, Y) \leftarrow \mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}.\textsc{Init}$ // Adversary D calls $\textsc{Init}$ of its own game

$\mathrm{I}^{\textsc{Init},\textsc{Fin}}$ // Run adversary I with subroutines below for its oracles

$\textsc{Init}()$: // Subroutine defined by D

Return $(1^\lambda, Y)$ // Reply to I's $\textsc{Init}$ query

$\textsc{Fin}(X')$: // Subroutine defined by D. Here X' is provided by I.

If $\mathsf{G}(1^\lambda, X') = Y$ then $b' \leftarrow 1$ else $b' \leftarrow 0$

$\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}.\textsc{Fin}(b')$ // Adversary D calls $\textsc{Fin}$ of its own game

Figure 2: Top: Games defining OW and PR security for generator G in the blended security setting. Bottom: Adversary D for Theorem 2.3.

meaning returns an X' such that $\mathsf{G}(X') = Y$, then b' equals 1, so we have

$$\Pr\left[\, b' = 1 \mid b = 1 \,\right] \geq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G}}(\mathrm{I}) \ .$$

Let $R = \{\, \mathsf{G}(X) \,:\, X \in \{0,1\}^{\mathsf{G}.\mathrm{il}} \,\} \subseteq \{0,1\}^{\mathsf{G}.\mathrm{ol}}$. If $b = 0$ then

$$\Pr\left[\, b' = 1 \mid b = 0 \,\right] \leq \Pr[Y \in R] = \frac{|R|}{2^{\mathsf{G}.\mathrm{ol}}} \leq \frac{2^{\mathsf{G}.\mathrm{il}}}{2^{\mathsf{G}.\mathrm{ol}}} \ .$$

Putting the above together completes the proof. ∎

2.3 Blended

A generator is a PT deterministic algorithm G that takes 1^λ and an input $X \in \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)}$ to return an output $\mathsf{G}(1^\lambda, X) \in \{0,1\}^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$. Here $\mathsf{G}.\mathrm{il}, \mathsf{G}.\mathrm{ol}\colon \mathbb{N} \to \mathbb{N}$ are PT deterministic functions determining the input and output lengths. The ow-advantage $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G},\mathrm{I}}(\cdot)\colon \mathbb{N} \to \mathbb{R}$ of an adversary I is defined by

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G},\mathrm{I}}(\lambda) = \Pr[\mathbf{G}^{\mathrm{ow}}_{\mathsf{G},\lambda}(\mathrm{I})]$$

for all $\lambda \in \mathbb{N}$, where game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{G},\lambda}$ is on the top left in Figure 2. We say that G is OW-secure if for every PT adversary I the function $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G},\mathrm{I}}(\cdot)$ is negligible. The pr-advantage $\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G},\mathrm{D}}(\cdot)\colon \mathbb{N} \to \mathbb{R}$ of an adversary D is defined by

$$\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G},\mathrm{D}}(\lambda) = 2\Pr[\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}(\mathrm{D})] - 1$$

for all $\lambda \in \mathbb{N}$, where game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}$ is on the top right in Figure 2. We say that G is PR-secure if for every PT adversary D the function $\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G},\mathrm{D}}(\cdot)$ is negligible.

Theorem 2.3 *Let* G *be a generator. Given an adversary* I *we can construct an adversary* D *(shown explicitly at the bottom of Figure 2) such that for all* $\lambda \in \mathbb{N}$ *we have*

$$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G},\mathrm{I}}(\lambda) \leq \mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G},\mathrm{D}}(\lambda) + \frac{1}{2^{\mathsf{G}.\mathrm{ol}(1^\lambda)-\mathsf{G}.\mathrm{il}(1^\lambda)}} \; . \tag{3}$$

The running time of D *is about the same as that of* I.

Proof of Theorem 2.3: Let $\lambda \in \mathbb{N}$. Let b be the challenge bit chosen at random in game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}$, and b' the bit queried by D to $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}.\textsc{Fin}$. Then (appealing here to Lemma 3.2) we have

$$\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G},\mathrm{D}}(\lambda) = \Pr\left[\, b'=1 \;|\; b=1 \,\right] - \Pr\left[\, b'=1 \;|\; b=0 \,\right]$$

where the probabilities are in the execution of game $\mathbf{G}^{\mathrm{pr}}_{\mathsf{G},\lambda}$ with adversary D. If $b=1$ and I succeeds, meaning returns an X' such that $\mathsf{G}(1^\lambda, X') = Y$, then b' equals 1, so we have

$$\Pr\left[\, b'=1 \;|\; b=1 \,\right] \geq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{G},\mathrm{I}}(\lambda) \; .$$

Let $R = \{\, \mathsf{G}(X) \,:\, X \in \{0,1\}^{\mathsf{G}.\mathrm{il}(1^\lambda)} \,\} \subseteq \{0,1\}^{\mathsf{G}.\mathrm{ol}(1^\lambda)}$. If $b=0$ then

$$\Pr\left[\, b'=1 \;|\; b=0 \,\right] \leq \Pr[Y \in R] = \frac{|R|}{2^{\mathsf{G}.\mathrm{ol}}} \leq \frac{2^{\mathsf{G}.\mathrm{il}(1^\lambda)}}{2^{\mathsf{G}.\mathrm{ol}(1^\lambda)}} \; .$$

Putting the above together completes the proof. ▮

Note that Theorem 2.1 is a corollary of Theorem 2.3, but not, at least formally, of Theorem 2.2.

2.4 Discussion

Schemes (here, G) have a different syntax in the asymptotic and concrete settings, taking input a security parameter (in unary) in the former, but not in the latter. In the concrete treatment, the security parameter does not exist.

Some real-world primitives, like AES or SHA256, do not have a security parameter and would not fit the asymptotic setting. The concrete setting covers them.

In the asymptotic setting, what it means for a scheme (here G) to be secure (here in the PR or OW sense) is formally well defined. In the concrete setting, it is not: we define the security metric via the adversary advantage function, but stop short of formally defining what it means for the scheme to be "secure" under this metric. When we say G is PR-secure, it is an informal statement, to be interpreted as $\mathbf{Adv}^{\mathrm{pr}}_{\mathsf{G}}(\mathrm{D})$ is "small" for all adversaries D whose resources are "practical." This is generally not a difficulty with simple notions like the ones in this example. But it makes more difficult the conceptual understanding of more complex notions like ZK which involve a particular quantification over different objects (adversary, simulator and so on) that, in the concrete setting, are all just parameters.

The asymptotic setting does not (usually) define any explicit advantage function. This makes it difficult (but not impossible) to make quantitative statements about the relationships between these advantages and (in my view) makes it harder than in the concrete setting to write precise proofs, as can be seen by comparing the proofs of Theorems 2.1 and 2.2. Of course, one *could* define advantage functions, and this is exactly what the blended setting does, but historically, and in canonical treatments [Gol01], this does not seem to be done.

The asymptotic setting does not (usually) use games, instead expressing adversary success probabilities directly. It tends to present adversaries in text rather than in pseudocode.

Theorems are formal statements in both settings. But the ones in the concrete setting explicitly state relations between adversary advantages and resources, and often even point to an explicit pseudocode adversary construction as the final determinant of its resource utilization. The running-time relation, however, tends to be a bit fuzzy for lack of a precise computational model, and we see statements, like in Theorem 2.2, about one running time being "about the same" as the other. Subtleties of which to be aware include code-size as an adversary resource and the consequences of the inherent non-uniformity of the setting [BL13].

Proofs in the asymptotic setting have tended, historically, to proceed by contradiction, with the template illustrated in the proof of Theorem 2.1, in contrast to proofs in the concrete setting, which proceed in the direct way illustrated by the proof of Theorem 2.2.

The notions we have considered here are simple. In asymptotic definitions for more complex notions, one can see many different parameters that must be appropriately quantified relative to something defined as being negligible. Asymptotic definitions of this type in the literature can be ambiguous, and even incorrect, as illustrated for example by [BH15]. The blended setting is a good remedy, forcing one to pin down the advantage function asked to be negligible.

The blended setting tries to be the "best of the two worlds." It has a security parameter, and does define adversary advantages, but, rather than numbers, these are now functions, of the security parameter. What it means for a scheme to be secure is formally well-defined, and identical to the asymptotic setting. However, theorems are able to state the relations between adversary advantages and resources as in the concrete setting. Subtleties as mentioned above are largely avoided. (It isn't perfect; for example, it does not directly capture AES and SHA256. But there are ways around this.)

The blended setting would be my choice for treating NIZKs, and what I would use in a paper on the subject. But for simplicity, these notes use the concrete setting.

3 Preliminaries

NOTATION. If $\boldsymbol{w}$ is a vector then $|\boldsymbol{w}|$ is its length (the number of its coordinates) and $\boldsymbol{w}[i]$ is its i-th coordinate. Strings are identified with vectors over $\{0,1\}$, so that $|Z|$ denotes the length of a string Z and $Z[i]$ denotes its i-th bit. By ε we denote the empty string or vector. By $x\|y$ we denote the concatenation of strings x, y. If x, y are equal-length strings then $x \oplus y$ denotes their bitwise xor. If S is a finite set, then $|S|$ denotes it size. We say that a set S is *length-closed* if, for any $x \in S$ it is the case that $\{0,1\}^{|x|} \subseteq S$. (This will be a requirement for message spaces.)

If X is a finite set, we let $x \leftarrow_{\$} X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;\omega)$ denote running A on inputs $x_1,\ldots$ and coins ω, with oracle access to $\mathrm{O}_1,\ldots$, and assigning the output to y. By $y \leftarrow_{\$} A^{\mathrm{O}_1,\cdots}(x_1,\ldots)$ we denote picking ω at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;\omega)$. We let $[A^{\mathrm{O}_1,\cdots}(x_1,\ldots)]$ denote the set of all possible outputs of A when run on inputs $x_1,\ldots$ and with oracle access to $\mathrm{O}_1,\ldots$. An adversary is an algorithm. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

GAMES. We use the code-based game-playing framework of BR [BR06]. A game G (see Figure 5 for examples) starts with an optional INIT procedure, followed by a non-negative number of additional procedures called oracles, and ends with a FIN procedure. Execution of adversary A with game G consists of running A with oracle access to the game procedures, with the restrictions that A's first call must be to INIT (if present), its last call must be to FIN, and it can call these procedures at most once. The output of the execution is the output of FIN. By $\Pr[\mathrm{G}(\mathrm{A}) \Rightarrow y]$ we denote the probability that the execution of game G with adversary A results in this output being y, and write just $\Pr[\mathrm{G}(\mathrm{A})]$ when $y = \mathsf{true}$. (Meaning $\Pr[\mathrm{G}(\mathrm{A})]$ is the probability that the execution of game G with adversary A results in the output of the execution being the boolean true.)

Note that our adversaries have no output. The role of what in other treatments is the adversary output is, for us, played by the query to FIN.

Different games may have procedures (oracles) with the same names. If we need to disambiguate, we may write G.O to refer to oracle O of game G.

In games, integer variables, set variables boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$.

The running time of an adversary executing with a game excludes the time taken by game procedures to compute answers to queries.

GAME-PLAYING LEMMAS. A flag is a boolean variable. Recall that any flag is initialized to false. If game G contains a flag bad then "G(A) sets bad" refers to the event that bad is set to true at some point in the execution of G with A. (As a clarification, this does not mean that bad is necessarily true when the game terminates.) We say that games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-bad if bad is a flag in both games and the games differ only in code following a statement $\mathsf{bad} \leftarrow \mathsf{true}$. The following is the Fundamental Lemma of Game Playing from [BR06].

Lemma 3.1 *Let $\mathrm{G}_0, \mathrm{G}_1$ be identical-until-bad games, and A an adversary. Then for all $b \in \{0,1\}$ and all y we have*

$$\Pr[\mathrm{G}_0(\mathrm{A}) \Rightarrow y] - \Pr[\mathrm{G}_1(\mathrm{A}) \Rightarrow y] \leq \Pr[\mathrm{G}_b(\mathrm{A}) \text{ sets } \mathsf{bad}] \; .$$

Also

$$\Pr[\mathrm{G}_0(\mathrm{A}) \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_1(\mathrm{A}) \text{ sets } \mathsf{bad}]$$

$$\Pr[\mathrm{G}_0(\mathrm{A}) \Rightarrow y \;\wedge\; \mathrm{G{\scriptsize D}}] = \Pr[\mathrm{G}_1(\mathrm{A}) \Rightarrow y \;\wedge\; \mathrm{G{\scriptsize D}}] \; ,$$

where GD *denotes the event that* bad *is never set to* true.

Above, for $b \in \{0,1\}$, event GD in the execution of G_b with A is the complement of the event "$\mathrm{G}_b(\mathrm{A})$ sets bad".

A decision problem is one where the adversary's task is to guess a challenge bit. One can formulate such a problem either via single game with the advantage defined as twice the probability of guessing the challenge bit minus one, or via two games, with the advantage defined as the difference in probabilities that the adversary's guess is 1, and these two formulations are equivalent. Since this equivalence is used often and for many problems, we try here to formalize a statement about it.

Let G be a game. We say that it is a *decision game* if the following hold. There is a boolean variable b (called the challenge bit) such that G.INIT includes the statement $b \leftarrow\!\!\$\; \{0,1\}$. This is the only code in the game that changes the value of b, so that the game in particular has no other statement assigning a value to b. The input to G.FIN includes a bit b' (call the guess bit), and G.FIN, via a statement "return $(b' = b)$" that is the only one in this procedure containing the "return" instruction, returns the boolean $(b' = b)$. This procedure has no statement changing the value of, or assigning a value to, b'. By G[0] we denote the G with the following modifications. The variable b set to 0. This means the statement $b \leftarrow\$ \{0,1\}$ is removed from INIT, and, for any reference to variable b in G, game G[0] uses the value 0. Oracle G[0].FIN takes the same inputs as G.FIN, including b', but replaces the "return $(b' = b)$" statement by "return $(b' = 1)$," meaning returns true iff b' is the bit 1. Game G[1] is defined correspondingly.

We warn that the above definitions are not entirely rigorous in the absence of a precise programming language [BR06]. However, we will apply the Lemma below only for explicitly-specified games G that we write in our definitions and proofs, and in these cases the conditions, and objects referred to, will be clear enough. But we warn that we are not yet at the stage of being able to formally and rigorously treat a game as an abstract mathematical object, and recommend caution in trying to doing this.

The following says that for decision games, the two ways of defining adversary advantage coincide.

Lemma 3.2 *Let* G *be a decision game, and* A *an adversary.*

$$2\Pr[\mathrm{G}(\mathrm{A})] - 1 = \Pr[\mathrm{G}[1](\mathrm{A})] - \Pr[\mathrm{G}[0](\mathrm{A})] \ .$$

Proof of Lemma 3.2: Let b be the challenge bit and b' the guess bit in G, and consider the execution of G with A. We have

$$\Pr\left[\, b' = 1 \mid b = 1 \,\right] = \Pr[\mathrm{G}[1](\mathrm{A})] \tag{4}$$

$$\Pr\left[\, b' = 1 \mid b = 0 \,\right] = \Pr[\mathrm{G}[0](\mathrm{A})] \ . \tag{5}$$

Given this we have

$$\begin{aligned}
2\Pr[\mathrm{G}(\mathrm{A})] - 1 &= 2\Pr[b' = b] - 1 \\
&= 2 \cdot \left(\Pr\left[\, b' = 1 \mid b = 1 \,\right] \cdot \Pr[b = 1] + \Pr\left[\, b' = 0 \mid b = 0 \,\right] \cdot \Pr[b = 0]\right) - 1 \\
&= 2 \cdot \left(\Pr\left[\, b' = 1 \mid b = 1 \,\right] \cdot \frac{1}{2} + \left(1 - \Pr\left[\, b' = 1 \mid b = 0 \,\right]\right) \cdot \frac{1}{2}\right) - 1 \\
&= 2 \cdot \left(\Pr\left[\, b' = 1 \mid b = 1 \,\right] \cdot \frac{1}{2} - \Pr\left[\, b' = 1 \mid b = 0 \,\right] \cdot \frac{1}{2} + \frac{1}{2}\right) - 1 \\
&= \Pr\left[\, b' = 1 \mid b = 1 \,\right] - \Pr\left[\, b' = 1 \mid b = 0 \,\right] \\
&= \Pr[\mathrm{G}[1](\mathrm{A})] - \Pr[\mathrm{G}[0](\mathrm{A})] \ ,
\end{aligned}$$

where the last equality uses Equations (4) and (5). ▮

REDUCTIONS. Proofs give reductions that take a G_2-adversary A_2 and specify (construct) a G_1-adversary A_1 that runs A_2 as a subroutine, itself responding to oracle queries of A_2. Let INIT, $\mathrm{O1}_1, \ldots, \mathrm{O1}_{n_1}$, FIN denote the oracles of G_1 and INIT, $\mathrm{O2}_1, \ldots, \mathrm{O2}_{n_2}$, FIN the oracles of G_2. Then we may write pseudocode of the form

Adversary A_1

$\vdots$

$\mathrm{A}_2^{\textsc{Init},\mathrm{O2}_1,\ldots,\mathrm{O2}_{n_2},\textsc{Fin}}$ // Run A_2 with specified subroutines as oracles

$\vdots$

$\textsc{Init}$ // Subroutine simulating $\mathrm{G}_2.\textsc{Init}$

$\vdots$

$\mathrm{O2}_1(\ldots)$ // Subroutine simulating $\mathrm{G}_2.\mathrm{O2}_1$

$\vdots$

$\textsc{Fin}(\ldots)$ // Subroutine simulating $\mathrm{G}_2.\textsc{Fin}$

Here $\textsc{Init}, \mathrm{O2}_1, \ldots, \mathrm{O2}_{n_2}, \textsc{Fin}$ are subroutines, given in the code of A_1, that are responsible for simulating the corresponding oracles for A_2 in G_2, and will invoke A_1's oracles to do so. These subroutines can invoke oracles from G_1, which, to disambiguate, will the given the game name as prefix, so that for example subroutine $\textsc{Init}$ can call $\mathrm{G}_1.\textsc{Init}$ and subroutine $\textsc{Fin}$ can call $\mathrm{G}_1.\textsc{Fin}$.

We adopt the convention that if a simulation is trivial, meaning $\mathrm{O2}_i(x)$ returns $\mathrm{O1}_j(x)$, then, in the superscripts to A_2, we simply write $\mathrm{O1}_j$ in place of $\mathrm{O2}_i$, and do not give code for the simulated oracle.

Circuits. We formalize circuits following [BHR12]. A *circuit* C specifies the following. The number of inputs is $\mathsf{C.In}$. There is only one output. There are $\mathsf{C.Gt}$ gates, numbered $\mathsf{C.In}+1, \ldots, \mathsf{C.In}+\mathsf{C.Gt}$. There are $\mathsf{C.In}+\mathsf{C.Gt}$ wires, numbered $1, \ldots, \mathsf{C.In}+\mathsf{C.Gt}$. Wires $1, \ldots, \mathsf{C.In}$ are the input wires, and wire $g \in [\mathsf{C.In}+1, \ldots, \mathsf{C.In}+\mathsf{C.Gt}]$ is the output wire of gate g. (Every non-input wire is the output wire of exactly one gate.) Functions $\mathsf{C.I1}, \mathsf{C.I2}$ take a gate $\mathtt{G} \in [\mathsf{C.In}+1..\mathsf{C.In}+\mathsf{C.Gt}]$ and return its first input wire $\mathsf{C.I1}(\mathtt{G})$ and its second input wire $\mathsf{C.I2}(\mathtt{G})$, satisfying $\mathsf{C.I1}(\mathtt{G}) < \mathsf{C.I2}(\mathtt{G}) < \mathtt{G}$. Gate $\mathsf{C.In}+\mathsf{C.Gt}$ is the output gate and wire $\mathsf{C.In}+\mathsf{C.Gt}$ is the output wire. There is only one type of gate, which is a 2-input NAND.

Function Ev evaluates a circuit. It takes a circuit C and a string $x \in \{0,1\}^{\mathsf{C.In}}$ giving assignments to the inputs of the circuit. It returns a vector $\boldsymbol{w}$ such that $\boldsymbol{w}[i]$ is the value of wire i. In particular, $\boldsymbol{w}[\mathsf{C.In}+\mathsf{C.Gt}]$ is the circuit output:

<u>Algorithm $\mathsf{Ev}(\mathsf{C}, x)$</u>
For $i = 1, \ldots, \mathsf{C.In}$ do $\boldsymbol{w}[i] \leftarrow x[i]$
For $i = \mathsf{C.In}+1, \ldots, \mathsf{C.Gt}$ do $\boldsymbol{w}[i] \leftarrow \mathrm{NAND}(\mathsf{C.I1}(i), \mathsf{C.I2}(i))$
Return $\boldsymbol{w}$

Function families. A function family $\mathsf{F}\colon \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \mathsf{F.R}$ takes a key K and input X to return output $\mathsf{F}(K, X)$, where $\mathsf{F.kl}$ is the key length and $\mathsf{F.D}$ is the domain. We require that $\mathsf{F}(K, X) = \bot$ if $X \notin \mathsf{F.D}$. We will consider several notions of security for function families.

By $\mathrm{FUNC}(D, R)$ we denote the set of all functions $f\colon D \to R$.

The one-wayness advantage of adversary A against F is defined by $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{ow}}_{\mathsf{F}}(\mathrm{A})]$ where the game is on the top left in Figure 3. This assumes $\mathsf{F.D}$ is finite. It is required that the X' queried by the adversary to $\textsc{Fin}$ be in $\mathsf{F.D}$.

We define PRF security for function family F via the game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$ on the top right in Figure 3. The

Game $\mathbf{G}^{\mathrm{ow}}_{\mathsf{F}}$	Game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$
$\textsc{Init}()$: $K \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$; $X \leftarrow\$ \mathsf{F.D}$; $Y \leftarrow \mathsf{F}(K,X)$ Return (K,Y) $\textsc{Fin}(X')$: Return ($\mathsf{F}(K,X') = Y$)	$\textsc{Init}()$: $b \leftarrow\$ \{0,1\}$ If $(b=1)$ then $K \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$; $f \leftarrow \mathsf{F}(K,\cdot)$ Else $f \leftarrow\$ \mathrm{FUNC}(\mathsf{F.D},\mathsf{F.R})$ $\textsc{Fn}(X)$: Return $f(X)$ $\textsc{Fin}(b')$: Return $(b'=b)$

Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}$

$\textsc{Init}()$:
$K \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$

$\textsc{Fn}(X)$:
$S \leftarrow S \cup \{X\}$; Return $\mathsf{F}(K,X)$

$\textsc{Fin}(X,T)$:
Return ($(\mathsf{F}(K,X) = T)$ and $(X \notin S)$)

Figure 3: Games defining OW and PRF security (top) and UF security (bottom) for function family F.

prf advantage of adversary A is $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(\mathrm{A})] - 1$. It is required that the X queried by the adversary to $\textsc{Fn}$ be in $\mathsf{F.D}$.

We define UF security for function family F via the game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}$ on the bottom in Figure 3. The uf advantage of adversary A is $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{F}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}(\mathrm{A})]$. It is required that the X queried by the adversary to $\textsc{Fn}$ or submmitted to $\textsc{Fin}$ be in $\mathsf{F.D}$. This captures unforgeability of F as a message authentication code.

Commitment schemes. A commitment scheme CS specifies a parameter-generation algorithm $\mathsf{CS.P}$ and a deterministic commitment algorithm $\mathsf{CS.C}$ such that $\mathsf{CS.C}(cp,\cdot,\cdot)\colon \mathsf{CS.M}(cp) \times \mathsf{CS.D}(cp) \to \{0,1\}^*$ for every $cp \in [\mathsf{CS.P}]$, where $\mathsf{CS.M}, \mathsf{CS.D}$, which are part of the scheme specification, associate to any choice cp of parameters a finite set $\mathsf{CS.M}(cp)$ of messages and a finite set $\mathsf{CS.D}(cp)$ of de-committment keys, respectively. We assume $[\mathsf{CS.P}]$ is finite too. The scheme is set up by generating parameters $cp \leftarrow\$ \mathsf{CS.P}$. Then via $c \leftarrow \mathsf{CS.C}(cp,m,d)$, one generates a commitment to message $m \in \mathsf{CS.M}(cp)$ with randomly-chosen de-commitment key $d \leftarrow\$ \mathsf{CS.D}(cp)$.

The bind advantage of adversary A is $\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{CS}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{bind}}_{\mathsf{CS}}(\mathrm{A})]$, where the game is in Figure 4. It is required that the m', d' queried by the adversary to $\textsc{Fin}$ satisfy $m' \in \mathsf{CS.M}(cp)$ and $d' \in \mathsf{CS.D}(cp)$. This captures a weak notion of binding where one of the messages is selected at random rather than by the adversary. We say that CS has perfect binding if $\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{CS}}(\mathrm{A}) = 0$ for all adversaries A. (All means regardless of their running time.)

The hide advantage of adversary A is $\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{CS}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{hide}}_{\mathsf{CS}}(\mathrm{A})] - 1$, where the game is in Figure 4. It is required that the messages m_0, m_1 queried by the adversary to CMT satisfy $m_0, m_1 \in \mathsf{CS.M}(cp)$. We say that CS has perfect hiding if $\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{CS}}(\mathrm{A}) = 0$ for all adversaries A. (All means

Game $\mathbf{G}^{\text{bind}}_{\mathsf{CS}}$

$\text{INIT}()$:
$cp \leftarrow\$ \mathsf{CS.P}$; $m \leftarrow\$ \mathsf{CS.M}(cp)$; $d \leftarrow\$ \mathsf{CS.D}(cp)$
Return (cp, m, d)

$\text{FIN}(m', d')$:
Return $(\,(\mathsf{CS.C}(cp, m', d') = \mathsf{CS.C}(cp, m, d))$ and $(m \neq m')\,)$

Game $\mathbf{G}^{\text{hide}}_{\mathsf{CS}}$

$\text{INIT}()$:
$cp \leftarrow\$ \mathsf{CS.P}$; $b \leftarrow\$ \{0,1\}$
Return cp

$\text{CMT}(m_0, m_1)$:
$d \leftarrow\$ \mathsf{CS.D}(cp)$
Return $\mathsf{CS.C}(cp, m_b, d)$

$\text{FIN}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\text{mode}}_{\mathsf{DCS}}$

$\text{INIT}()$:
$\mu \leftarrow\$ \{0,1\}$; $cp \leftarrow\$ \mathsf{DCS}_{\mu}.\mathsf{P}$; Return cp

$\text{FIN}(\mu')$:
Return $(\mu' = \mu)$

Figure 4: Top: Games defining BIND and HIDE security for commitment scheme CS. Bottom: Game defining mode-indistinguishability of dual-mode commitment scheme DCS.

---

regardless of their running time and the number of CMT queries they make.)

If CS is a commitment scheme, $cp \in [\mathsf{CS.P}]$ and $m \in \mathsf{CS.M}(cp)$ then it is useful to let

$$\mathsf{CS.C}(cp, m) = \{\, \mathsf{CS.C}(cp, m, d) \;:\; d \in \mathsf{CS.D}(cp) \,\}$$

be the set of all possible commitments of message m relative to cp. You may find this definition useful in Exercise 3.3.

Exercise 3.3 *Prove that no commitment scheme has both perfect binding and perfect hiding.*

DUAL-MODE COMMITMENT SCHEMES. We introduce these, which are new as far as we know. A dual-mode commitment scheme DCS, like a commitment scheme, specifies a parameter-generation algorithm $\mathsf{DCS.P}$ and a deterministic commitment algorithm $\mathsf{DCS.C}$. The difference is that $\mathsf{DCS.P}$ takes an input $\mu \in \{0,1\}$ called the mode. The commitment algorithm maps $\mathsf{DCS.C}(cp, \cdot, \cdot)\colon \mathsf{DCS.M}(cp) \times \mathsf{DCS.D}(cp) \to \{0,1\}^*$ for every $cp \in [\mathsf{DCS.P}(0)] \cup [\mathsf{DCS.P}(1)]$. The scheme is set up in mode μ by generating parameters $cp \leftarrow\$ \mathsf{DCS.P}(\mu)$. Then via $c \leftarrow \mathsf{DCS.C}(cp, m, d)$, one generates a commitment to message $m \in \mathsf{DCS.M}(cp)$ with randomly-chosen de-commitment key $d \leftarrow\$ \mathsf{DCS.D}(cp)$, just like in a commitment scheme.

A dual-mode commitment scheme DCS gives rise to two (standard) commitment schemes that we call *the commitment schemes induced by* DCS and denote DCS_1 and DCS_0. Their commitment algorithms are that of DCS, meaning $\mathsf{DCS}_{\mu}.\mathsf{C} = \mathsf{DCS.C}$ for both $\mu \in \{0,1\}$. The difference between the two commitment schemes is in their parameter generation algorithms. Namely $\mathsf{DCS}_{\mu}.\mathsf{P}$ is defined by: $cp \leftarrow\$ \mathsf{DCS.P}(\mu)$; Return cp. The definitions we have above for commitment schemes now apply, or can be used for, either DCS_1 or DCS_0.

The value of dual-mode commitment schemes emerges when the parameters created in the two modes are indistinguishable. To formalize this, consider game $\mathbf{G}^{\mathrm{mode}}_{\mathsf{DCS}}$ of Figure 4 associated to dual-mode commitment scheme DCS, and let the mode advantage of adversary A be defined by $\mathbf{Adv}^{\mathrm{mode}}_{\mathsf{DCS}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{mode}}_{\mathsf{DCS}}(\mathrm{A})] - 1$.

Exercise 3.4 *Let* DCS *be a dual-mode commitment scheme and let* $\mu \in \{0,1\}$*. Formulate and prove results (theorems) saying the following: (1) If* DCS *is mode-indistinguishable and* DCS_μ *is binding-secure then* $\mathsf{DCS}_{1-\mu}$ *is binding-secure. (2) If* DCS *is mode-indistinguishable and* DCS_μ *is hiding-secure then* $\mathsf{DCS}_{1-\mu}$ *is hiding-secure.*

4 NIZK definitions and basic relations

A proof system provides a way for one party (the prover) to prove some "claim" to another party (the verifier). A claim is defined via a claim statement, which is a string x, and a claim validator CV, which is an algorithm that, with the help of a witness string w, says whether or not the claim statement is valid. What we call a claim validator is, in the literature, often called the relation, the claim statement being the input or theorem.

We start by formalizing claim validators, the associated set of true claims, and the syntax of proof systems. We formalize security properties of proof systems in three clusters. First is basic properties: soundness, extractability and ZK. Second is simulation soundness and extractability. Third is joint notions where a single game asks simultaneously for ZK and either soundness or extractability.

A few elements here seem new. One is the joint definitions, which facilitate applications, allowing a few hybrid proof steps to be compressed into one. Another is allowing multiple verification (for soundness) or extraction (for extractability) queries via an oracle for this purpose. (Conventional treatments, in this language, correspond to allowing one query.) This does not change the notion asymptotically, but it changes the concrete security, and allows us to see how constructions differ with regard to the way adversary advantage grows as a function of the number of queries to these oracles. We also introduce dual-mode NIZKs as a way to formalize something implicit in the literature.

These are not the only definitions. The literature considers many more.

4.1 Claim validators and proof systems

Claim validators. A *claim validator* is a function $\mathsf{CV}\colon \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$ that takes parameters pars, a claim statement x and a candidate witness w to return either true (saying w is a valid witness establishing the claim) or false (the witness fails to validate the claim). Common examples arise from the relations underlying **NP** languages. For example, x could be a circuit, w an assignment to its variables, and CV returns true iff w satisfies x. (In this example, $\mathrm{pars} = \varepsilon$.) For $\mathrm{pars}, x \in \{0,1\}^*$ we let $\mathsf{CV}(\mathrm{pars}, x) = \{\, w : \mathsf{CV}(\mathrm{pars}, x, w) = \mathsf{true} \,\}$ be the *witness set* of x.

A *true-claim language* is a function $\mathsf{TC}\colon \{0,1\}^* \to 2^{\{0,1\}^*}$, meaning it associates to any $\mathrm{pars} \in \{0,1\}^*$ a set $\mathsf{TC}(\mathrm{pars}) \subseteq \{0,1\}^*$. A claim validator CV gives rise to the true-claim language $\mathsf{TrCl}_{\mathsf{CV}}$

defined by $\mathsf{TrCl}_{\mathsf{CV}}(\mathrm{pars}) = \{\, x : \mathsf{CV}(\mathrm{pars}, x) \neq \emptyset \,\}$ called the *true-claim language associated to* CV. It associates to pars the set of valid (true) claim statements.

In usage of the proof system, the prover and verifier both know pars, x. (This situation may be arrived at in many ways, including the prover supplying x while pars being trusted parameters.) The prover is claiming that $x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{pars})$, and wants to establish this claim without revealing a witness w satisfying $\mathsf{CV}(\mathrm{pars}, x, w) = \mathsf{true}$ or, for that matter, anything else.

In the literature, CV often does not take pars as a separate input.

Proof systems. A proof system is the name of the syntax for the primitive that enables the production and verification of such proofs. Soundness, zero-knowledge and many other things will be security metrics for this primitive.

Proceeding, a *proof system* Π specifies the following algorithms:

- *CRS generation.* Via $\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$, the crs-generation algorithm $\Pi.\mathsf{C}$ (takes no inputs and) returns an output crs called the common reference string.
- *Proof generation.* Via $\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, x, w)$ the proof generation algorithm $\Pi.\mathsf{P}$ takes crs, a claim x and a witness w to produce a proof string. As this indicates the common reference string plays the role of the parameters.
- *Proof verification.* Via $d \leftarrow \Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf})$ the proof verification algorithm $\Pi.\mathsf{V}$ produces a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether or not it considers pf valid.

In some treatments, a proof system Π is defined as being for a claim validator CV. Our syntax views Π independently of CV. The claim validator shows up in defining attributes of Π below.

Completeness. We require perfect completeness, although this can be relaxed if necessary. We say that Π has (perfect) completeness for CV if $\Pi.\mathsf{V}(\mathrm{crs}, x, \Pi.\mathsf{P}(\mathrm{crs}, x, w)) = \mathsf{true}$ for all $\mathrm{crs} \in [\Pi.\mathsf{C}]$, all $x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$ and all $w \in \mathsf{CV}(\mathrm{crs}, x)$.

4.2 ZK and WI

Zero knowledge. Zero knowledge (ZK) of Π for CV asks that there be a simulator that, given a true statement (meaning, one in $\mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$) can create for it a proof indistinguishable from one an honest prover would provide, based, again, on a trapdoor underlying crs. The formalization has evolved over time. What we formalize is adaptive zero knowledge. A *simulator* is an object that specifies algorithms $\mathsf{S.C}$ (the simulation CRS-generator) and $\mathsf{S.P}$ (the simulation proof-generator). Consider game $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}$ specified in Figure 5. Adversary A can adaptively request proofs by supplying an instance and a valid witness for it. The proof is produced either by the honest prover using the witness, or by the proof simulator $\mathsf{S.P}$ using a trapdoor td. The adversary outputs a guess b' as to whether the proofs were real or simulated. Let $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})] - 1$.

In the asymptotic setting, Π is said to be zero-knowledge if there exists a polynomial-time simulator S such that $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 0$ for all A (perfect), $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})$ is negligible for all A (statistical) or $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})$ is negligible for all polynomial-time A (computational). In the concrete setting, the requirement may be understood as asking that there is an "efficient" simulator S such that $\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 0$ for all A (perfect), is "small" for all A (statistical) or is "small" for all practical

Game $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}$

INIT():
$b \leftarrow\!\!\$\ \{0,1\}$
If $(b = 1)$ then crs $\leftarrow\!\!\$\ \Pi.\mathsf{C}$
Else (crs, td) $\leftarrow\!\!\$\ \mathsf{S.C}$
Return crs

PF(x, w):
If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\perp$
If $(b = 1)$ then pf $\leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, x, w)$
Else pf $\leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$
Return pf

FIN(b'):
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}$

INIT():
crs $\leftarrow\!\!\$\ \Pi.\mathsf{C}$; $b \leftarrow\!\!\$\ \{0,1\}$
Return crs

PF(x, w_0, w_1):
If ($(\mathsf{CV}(\mathrm{crs}, x, w_0) = \mathsf{false})$ or $(\mathsf{CV}(\mathrm{crs}, x, w_1) = \mathsf{false})$)
  then return $\perp$
pf $\leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, x, w_b)$
Return pf

FIN(b'):
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}$

INIT():
crs $\leftarrow\!\!\$\ \Pi.\mathsf{C}$; Return crs

VF(x, pf):
If $(x \in \mathsf{TC}(\mathrm{crs}))$ then return false
If $(\Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf}))$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf})$

FIN():
Return win

Game $\mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}$

INIT():
(crs, td) $\leftarrow\!\!\$\ \mathsf{S.C}$; Return crs

EX(x, pf):
If $(\Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false
$w \leftarrow\!\!\$\ \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, x, \mathrm{pf})$
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{CV}(\mathrm{crs}, x, w)$

FIN():
Return win

Figure 5: Top: Games defining zero-knowledge (relative to simulator S) and witness indistinguishability of proof system Π. Bottom: Games defining soundness and extractability of proof system Π.

A (computational). Again, a more fine-grained classification becomes possible.

WITNESS INDISTINGUISHABILITY. This asks that, knowing $x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$ and knowing two witnesses $w_0, w_1 \in \mathsf{CV}(\mathrm{crs}, x)$, it is hard to tell under which of the two a proof has been computed. Consider game $\mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}$ specified in Figure 5. Let $\mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A})] - 1$.

ZK IMPLIES WI. The following says that ZK implies WI regardless of the running time of the simulator. (That the latter does not matter is reflected in the running time of A_{zk} not depending on the running time of the algorithms of S.)

Proposition 4.1 [ZK $\Rightarrow$ WI] *Let CV be a claim validator, Π a proof system, and S a simulator. Let A_{wi} be an adversary. Then we can construct adversary A_{zk} (shown explicitly in Figure 6) such that*

$$\mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}_{\mathrm{wi}}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) .$$

Adversary A_{zk} makes the same number of PF *queries as A_{wi} and its running time is about that of*

Adversary A_{zk}:

crs $\leftarrow \mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Init}$; $d \leftarrow\!\!\$ \{0,1\}$; $\mathrm{A}_{\mathrm{wi}}^{\textsc{Init},\textsc{Pf},\textsc{Fin}}$

$\textsc{Init}()$:

Return crs

$\textsc{Pf}(x, w_0, w_1)$:

If (($\mathsf{CV}(\mathrm{crs}, x, w_0) = \mathsf{false}$) or ($\mathsf{CV}(\mathrm{crs}, x, w_1) = \mathsf{false}$)) then return $\perp$

pf $\leftarrow\!\!\$ \mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Pf}(x, w_d)$; Return pf

$\textsc{Fin}(d')$:

If $(d = d')$ then $b' \leftarrow 1$ else $b' \leftarrow 0$

$\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(b')$

Figure 6: Adversary for Proposition 4.1.

A_{wi} *plus the time for two invocations of* CV*.*

Proof of Proposition 4.1: Let b denote the challenge bit in the execution of A_{zk} with game $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}$. Then

$$\Pr\left[\, b' = 1 \mid b = 1 \,\right] = \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}_{\mathrm{wi}})$$

$$\Pr\left[\, b' = 1 \mid b = 0 \,\right] = \frac{1}{2}\,.$$

By Lemma 3.2 we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) &= \Pr\left[\, b' = 1 \mid b = 1 \,\right] - \Pr\left[\, b' = 1 \mid b = 0 \,\right] \\ &= \frac{1}{2} \cdot \mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}_{\mathrm{wi}})\,.\end{aligned}$$

Make sure you understand why the different equations are true. We omit the details. ∎

4.3 Basic soundness and extractability: SND1 and XT1

SND1 SOUNDNESS. Soundness of Π is defined relative to a true-claim language TC which may then be set to $\mathsf{TrCl}_{\mathsf{CV}}$ for some choice of CV. It asks that it be hard to create a valid proof for $x \notin \mathsf{TC}(\mathrm{crs})$. Our formalization is via game $\mathbf{G}^{\mathrm{snd1}}$ in Figure 5. The adversary gets the common reference string crs. Then, it can submit to $\textsc{Vf}$ a statement x of its choice and a candidate proof pf for it, winning if $x \notin \mathsf{TC}(\mathrm{crs})$ yet the verification algorithm accepts pf as valid. It may call the oracle multiple times, as often as it wants. We let $\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A})]$ be its snd1-advantage.

The usual (asymptotic) definitions consider only one pair x, pf submitted by the adversary, which in our definition corresponds to limiting attention to adversaries that make a single $\textsc{Vf}$ query. Proposition 4.11 shows, via the simple and expected hybrid argument, that an adversary making q queries has an advantage at most q times that of an adversary of comparable resources making just one query. In the asymptotic security mindset, this justifies not considering multiple queries.

In the concrete security mindset, one goes further, asking, whether, for particular proof systems, there is a better reduction or result that avoids this factor q degradation. We will see that, for many proof systems, this is possible.

In the asymptotic setting, one speaks of soundness of Π being perfect ($\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}) = 0$ for all A), statistical ($\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A})$ is negligible for all A) or computational ($\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A})$ is negligible for all polynomial-time A). In the concrete setting, perfect soundness is still well-defined. Statistical would be understood as $\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A})$ being "small" for all A, and computational as it being "small" for all "practical" A. But in the concrete setting, a more fine-grained differentiation of levels of soundness is possible, in which statistical and computational are ends of a spectrum with much in between.

XT1 EXTRACTABILITY. The notion of Π being a proof of knowledge [GMR89, BG93, DP92] for CV requires that whenever a (potentially cheating) prover, modeled as the adversary, is able to produce a valid proof, there is an extractor that, based on a trapdoor underlying the common reference string, can extract the witness from the information available to the adversary. Our formalization is via game $\mathbf{G}^{\mathrm{xt1}}$ specified in Figure 5. It is parameterized by an *extractor* S, an object that specifies algorithms $\mathsf{S.C}$ (the extraction-CRS generator) and $\mathsf{S.X}$ (the extraction witness-generator). Let $\mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})]$ be the xt1-advantage of A.

The asymptotic definition of Π being XT1-secure would be that there exists a polynomial time extractor S such that $\mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})$ is negligible for all polynomial time adversaries A. In the concrete setting, the requirement may be understood as asking that there is an "efficient" extractor S such that $\mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})$ is "small" for all practical adversaries A.

Exercise 4.2 *Prove or disprove each of the following: (1) SND1 implies XT1 (2) XT1 implies SND1 (3) XT1+ZK (where the simulator is the same across the two) implies SND1.*

4.4 Simulation soundness and extractability: SND2/3, XT2/3

Simulation soundness [Sah99] asks that soundness hold under the simulated CRS, even when the soundness adversary can obtain proofs under the simulation trapdoor. SND2, the weaker version, restricts the proofs to ones on true statements, while SND3, the stronger version, drops this restriction.

SND2/3 SOUNDNESS. This requires that it is hard to create a valid proof for $x \notin \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{pars})$ even after seeing simulated proofs on claims of the adversary's choosing. Our formalization considers games $\mathbf{G}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}$, $\mathbf{G}^{\mathrm{snd3}}_{\Pi,\mathsf{CV},\mathsf{S}}$ shown in Figure 7. Here S, again, is a simulator, specifying algorithms $\mathsf{S.C}$ and $\mathsf{S.P}$. Line 2 is included only in $\mathbf{G}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}$, so that this game restricts PF to provide proofs only for valid claims, a restriction dropped in $\mathbf{G}^{\mathrm{snd3}}_{\Pi,\mathsf{CV},\mathsf{S}}$. (The witness input is thus pointless in the latter, and thus removed.) Game $\mathbf{G}^{\mathrm{snd3}}_{\Pi,\mathsf{CV},\mathsf{S}}$ adds lines 4,6 to exclude winning by providing proofs returned by PF. We let $\mathbf{Adv}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})]$ and $\mathbf{Adv}^{\mathrm{snd3}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{snd3}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})]$.

Exercise 4.3 *Consider a definition of soundness, call it SND4, that is like SND2 except that line 3 is replaced with* $\mathrm{pf} \leftarrow\!\$\; \Pi.\mathsf{P}(\mathrm{crs}, x, w)$. *Explore the relation of this to SND1, paying attention to the concrete security.*

Games $\mathbf{G}^{\mathrm{snd2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$, $\mathbf{G}^{\mathrm{snd3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$

$\text{I}\textsc{nit}()$:
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\; \mathsf{S.C}$; Return crs

$\text{P}\textsc{f}(x, \boxed{w})$: // Input w only in game $\mathbf{G}^{\mathrm{snd2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then return $\perp$ // Game $\mathbf{G}^{\mathrm{snd2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
$\mathrm{pf} \leftarrow\!\!\$\; \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$
$Q \leftarrow Q \cup \{(x, \mathrm{pf})\}$ // Game $\mathbf{G}^{\mathrm{snd3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
Return pf

$\text{V}\textsc{f}(x, \mathrm{pf})$:
If $((x, \mathrm{pf}) \in Q)$ then return false // Game $\mathbf{G}^{\mathrm{snd3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
If $(x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs}))$ then return false
If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}))$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$

$\text{F}\textsc{in}()$:
Return win

Games $\mathbf{G}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$, $\mathbf{G}^{\mathrm{xt3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$

$\text{I}\textsc{nit}()$:
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\; \mathsf{S.C}$; Return crs

$\text{P}\textsc{f}(x, \boxed{w})$: // Input w only in game $\mathbf{G}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then return $\perp$ // Game $\mathbf{G}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$
$\mathrm{pf} \leftarrow\!\!\$\; \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$; $Q \leftarrow Q \cup \{(x, \mathrm{pf})\}$
Return pf

$\text{E}\textsc{x}(x, \mathrm{pf})$:
If $((x, \mathrm{pf}) \in Q)$ then return false
If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false
$w \leftarrow\!\!\$\; \mathsf{S.X}(\mathrm{crs}_0, \mathrm{td}, x, \mathrm{pf})$
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{CV}(\mathrm{crs}, x, w)$

$\text{F}\textsc{in}()$:
Return win

Figure 7: Games defining SND2 soundness and XT2 extractability of proof system $\mathsf{\Pi}$.

XT2/3 EXTRACTABILITY. This requires that one be able to extract a witness underling a valid proof even when the claim was created after seeing simulated proofs. Our formalization considers games $\mathbf{G}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$, $\mathbf{G}^{\mathrm{xt3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}$ shown in Figure 7. Here S, the simulation extractor, specifies algorithms $\mathsf{S.C}$, $\mathsf{S.P}$ and $\mathsf{S.X}$. We let $\mathbf{Adv}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{xt2}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}(\mathrm{A})]$ and $\mathbf{Adv}^{\mathrm{xt3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{xt3}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}(\mathrm{A})]$.

Exercise 4.4 *Prove or disprove each of the following. (1) SND2+ZK implies SND1 (2) SND1+ZK implies SND2 (3) XT2+ZK implies XT1 (4) XT1+ZK implies XT2.*

Game G_0

$\textsc{Init}()$:
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$; Return crs

$\textsc{Pf}(x, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\bot$
$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$; Return pf

$\textsc{Vf}(x, \mathrm{pf})$:
If $(x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs}))$ then return false
If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}))$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$

$\textsc{Fin}$:
Return win

Game G_1

$\textsc{Init}()$:
$\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$; Return crs

$\textsc{Pf}(x, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\bot$
$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{\Pi.P}(\mathrm{crs}, x, w)$; Return pf

$\textsc{Vf}(x, \mathrm{pf})$:
If $(x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs}))$ then return false
If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}))$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$

$\textsc{Fin}$:
Return win

Adversary A_{zk}:

$\mathrm{crs} \leftarrow \mathbf{G}^{\mathrm{zk}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\textsc{Init}$; $\mathrm{A}_{\mathrm{snd2}}^{\textsc{Init},\textsc{Pf},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}()$:
Return crs

$\textsc{Pf}(x, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\bot$
$\mathrm{pf} \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{zk}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\textsc{Pf}(x, w)$; Return pf

$\textsc{Vf}(x, \mathrm{pf})$:
If $(x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs}))$ then return false
If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}))$ then $\mathsf{win} \leftarrow \mathsf{true}$
Return $\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$

$\textsc{Fin}$:
If (win) then $b' \leftarrow 0$ else $b' \leftarrow 1$
$\mathbf{G}^{\mathrm{zk}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\textsc{Fin}(b')$

Adversary $\mathrm{A}_{\mathrm{snd1}}$:

$\mathrm{crs} \leftarrow \mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Init}$; $\mathrm{A}_{\mathrm{snd2}}^{\textsc{Init},\textsc{Pf},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}()$:
Return crs

$\textsc{Pf}(x, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\bot$
$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{\Pi.P}(\mathrm{crs}, x, w)$; Return pf

$\textsc{Vf}(x, \mathrm{pf})$:
Return $\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Vf}$

$\textsc{Fin}$:
$\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Fin}$

Figure 8: Games and adversaries for Theorem 4.5.

Does SND1+ZK imply SND2? (The simulator is maintained across the question. So, given a proof system $\mathsf{\Pi}$ that is SND1, and is ZK relative to S, we are asking if it is SND2 relative to S.) It is tempting to think the answer is yes. Intuitively, if soundness is violated in the SND2 game but not in the SND1 game, it provides a way to distinguish between real and simulated proofs, violating ZK. The difficulty is that testing whether or not soundness is violated requires testing membership in the language of true claims, which is in general not doable in polynomial time. (Indeed, it is not in the case of most interest, where the language is NP-complete.) However a partial implication is still possible. Theorem 4.5 says that SND1+ZK does imply SND2 when the ZK is perfect, but Proposition 4.6 gives a counter-example to show that in general, the implication fails.

In Theorem 4.5 below, the running time of the constructed adversary A_{zk} depends on the time to test membership in $\mathsf{TrCl}_{\mathsf{CV}}$. The advantage of this adversary is thus small only when the zero-knowledge is of a strong form, like perfect. In contrast, adversary $\mathrm{A}_{\mathrm{snd1}}$ stays efficient, of running time about that of the given adversary $\mathrm{A}_{\mathrm{snd2}}$.

Claim validator $\mathsf{CV}(\mathrm{crs}, X, w)$:

$(A, B, C) \leftarrow \mathrm{crs}$
Return $(\,(g^w = A)$ and $(X \neq B^w)\,)$

CRS generator $\Pi.\mathsf{C}$:

$a, b \leftarrow\$\, \mathbb{Z}_p$; $c \leftarrow\$\, \mathbb{Z}_p$
$A \leftarrow g^a$; $B \leftarrow g^b$; $C \leftarrow g^c$
$\mathrm{crs} \leftarrow (A, B, C)$; Return crs

Proof generator $\Pi.\mathsf{P}(\mathrm{crs}, X, w)$:

Return ε

Proof verifier $\Pi.\mathsf{V}(\mathrm{crs}, X, \mathrm{pf})$:

Return true

Sim CRS generator $\mathsf{S.C}$:

$a, b \leftarrow\$\, \mathbb{Z}_p$; $c \leftarrow ab$
$A \leftarrow g^a$; $B \leftarrow g^b$; $C \leftarrow g^c$
$\mathrm{crs} \leftarrow (A, B, C)$; Return $(\mathrm{crs}, \varepsilon)$

Sim proof generator $\mathsf{S.P}(\mathrm{crs}, \mathrm{td}, X)$:

Return ε

Figure 9: Claim validator, proof system and simulator for Proposition 4.6.

Theorem 4.5 *Let* CV *be a claim validator,* Π *a proof system, and* S *a simulator. Let* $\mathrm{A}_{\mathrm{snd2}}$ *be an adversary. Then we can construct adversaries* $\mathrm{A}_{\mathrm{zk}}, \mathrm{A}_{\mathrm{snd1}}$ *(shown explicitly in Figure 8) such that*

$$\mathbf{Adv}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{snd2}}) \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}}) + \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) \; .$$

Suppose $\mathrm{A}_{\mathrm{snd2}}$ *makes* q_p *queries to its* $\mathrm{P}\scriptstyle{\mathrm{F}}$ *oracle and* q_v *queries to its* $\mathrm{V}\scriptstyle{\mathrm{F}}$ *oracle. Then adversary* A_{zk} *makes* q_p *queries to its* $\mathrm{P}\scriptstyle{\mathrm{F}}$ *oracle and its running time is about that of* $\mathrm{A}_{\mathrm{snd2}}$ *plus the time for* q_v *membership tests in* $\mathsf{TrCl}_{\mathsf{CV}}$*. Adversary* $\mathrm{A}_{\mathrm{snd1}}$ *makes* q_v *queries to its* $\mathrm{V}\scriptstyle{\mathrm{F}}$ *and its running time is about that of* $\mathrm{A}_{\mathrm{snd2}}$ *plus the time for* q_p *executions of* CV *and* $\Pi.\mathsf{P}$*.*

Proof of Theorem 4.5: Considering Figure 8, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{snd2}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{snd2}}) &= \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{snd2}})] \\ &= \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd2}})] + \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{snd2}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd2}})] \; .\end{aligned}$$

To complete the proof we claim that

$$\begin{aligned}\Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd2}}] &\leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}}) \\ \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{snd2}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd2}})] &\leq \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) \; .\end{aligned}$$

The details are left as an exercise. ▌

The following says that SND1+ZK does not in general imply SND2. It assumes that DDH (and hence CDH) is hard in the group $\mathbb{G}$.

Proposition 4.6 *Let* $\mathbb{G}$ *be a group of order* p *and let* $g \in \mathrm{Gen}(\mathbb{G})$*. Let* CV *be the claim validator,* Π *the proof system, and* S *the simulator, of Figure 9.*

1. Π *is ZK for* CV *relative to* S*, assuming DDH:*

2. Π *is SND1 for* $\mathsf{TrCl}_{\mathsf{CV}}$*, assuming CDH:*

3. Π *is not SND2 for* $\mathsf{TrCl}_{\mathsf{CV}}$ *relative to* S*:*

Game $\mathbf{G}^{\mathrm{zksnd}}_{\Pi,\mathsf{CV},\mathsf{S}}$

$\textsc{Init}()$:
$b \leftarrow\$ \{0,1\}$
If $(b = 1)$ then $\mathrm{crs} \leftarrow\$ \Pi.\mathsf{C}$
Else $(\mathrm{crs}, \mathrm{td}) \leftarrow\$ \mathsf{S.C}$
Return crs

$\textsc{Pf}(x, w)$:
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then return $\perp$
If $(b = 1)$ then $\mathrm{pf} \leftarrow\$ \Pi.\mathsf{P}(\mathrm{crs}, x, w)$
Else $\mathrm{pf} \leftarrow\$ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$
Return pf

$\textsc{Vf}(x, \mathrm{pf})$:
If $(x \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{pars})))$ then return false
If $(b = 0)$ then return false
Return $\Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf})$

$\textsc{Fin}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}$

$\textsc{Init}()$:
$b \leftarrow\$ \{0,1\}$
If $(b = 1)$ then $\mathrm{crs} \leftarrow\$ \Pi.\mathsf{C}$
Else $(\mathrm{crs}, \mathrm{td}) \leftarrow\$ \mathsf{S.C}$
Return crs

$\textsc{Pf}(x, w)$:
If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then return $\perp$
If $(b = 1)$ then $\mathrm{pf} \leftarrow\$ \Pi.\mathsf{P}(\mathrm{crs}, x, w)$
Else $\mathrm{pf} \leftarrow\$ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$
$Q \leftarrow Q \cup \{(x, \mathrm{pf})\}$
Return pf

$\textsc{Ex}(x, \mathrm{pf})$:
If $((x, \mathrm{pf}) \in Q)$ then return false
If $(\Pi.\mathsf{V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false
If $(b = 1)$ then return true
$w \leftarrow\$ \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, x, \mathrm{pf})$
Return $\mathsf{CV}(\mathrm{crs}, x, w)$

$\textsc{Fin}(b')$:
Return $(b' = b)$

Figure 10: Games defining joint ZK + soundness/extractability of proof system Π.

4.5 Joint notions: ZKSND, ZKXT

We give some new definitions where ZK and soundness/extractability are captured in a single game. This makes proofs of applications easier. Let $\mathbf{Adv}^{\mathrm{zksnd}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{zksnd}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})] - 1$ and $\mathbf{Adv}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})] - 1$ where the games are in Figure 10.

Exercise 4.7 *Prove or disprove each of the following. (1) SND2+ZK implies ZKSND (2) ZKSND implies SND2+ZK (3) XT2+ZK implies ZKXT (4) ZKXT implies XT2+ZK.*

4.6 Dual-mode systems

We are going to experiment with an additional syntax that I call a *dual-mode proof system*. (It is new to best of my knowledge, but I'd appreciate a literature check or feedback in that regard.) A dual-mode proof system $\mathsf{D\Pi}$ specifies a proof generation algorithm $\mathsf{D\Pi.P}$ and a proof verification algorithm $\mathsf{D\Pi.V}$ that are just like those of a proof system in syntax. The difference is the CRS generator $\mathsf{D\Pi.C}$, which now takes an input $\mu \in \{0,1\}$ called the *mode* and returns a pair $(\mathrm{crs}, \mathrm{td})$ consisting of a common reference string and a trapdoor.

A dual-mode proof system $\mathsf{D\Pi}$ gives rise to two (standard) proof systems that we call *the proof systems induced by* $\mathsf{D\Pi}$ and denote Π_1 and Π_0. Their proof generation and verification algorithms are those of $\mathsf{D\Pi}$, meaning $\Pi_\mu.\mathsf{P} = \mathsf{D\Pi.P}$ and $\Pi_\mu.\mathsf{V} = \mathsf{D\Pi.V}$, for both $\mu \in \{0,1\}$. The difference between the two proof systems is in their CRS generation algorithms. Namely $\Pi_\mu.\mathsf{C}$ is defined by:

Game $\mathbf{G}^{\mathrm{mode}}_{\mathsf{D\Pi}}$

$\text{Init}()$:

$\mu \leftarrow\!\!\$\ \{0,1\}$; $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{D\Pi.C}(\mu)$; Return crs

$\text{Fin}(\mu')$:

Return $(\mu' = \mu)$

Adversary $\mathrm{A}_{\mathrm{mode}}$:

$\mathrm{crs} \leftarrow \mathbf{G}^{\mathrm{mode}}_{\mathsf{D\Pi}}.\text{Init}$; $\mathrm{A}_{\mathrm{zk}}^{\text{Init},\text{Pf},\text{Fin}}$

$\text{Init}()$:

Return crs

$\text{Pf}(x, w)$:

If (not $\mathsf{CV}(\mathrm{crs}, x, w)$) then return $\perp$

$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{D\Pi.P}(\mathrm{crs}, x, w)$; Return pf

$\text{Fin}(b')$:

$\mathbf{G}^{\mathrm{mode}}_{\mathsf{D\Pi}}.\text{Fin}(b' \oplus \mu)$

Figure 11: Top: Game defining mode indistinguishability for a dual-mode proof system $\mathsf{D\Pi}$. Bottom: Mode-indistinguishability adversary for Thereom 4.8.

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{D\Pi.C}(\mu)$; Return crs. The definitions we have above for proof systems now apply, or can be used for, either $\mathsf{\Pi}_1$ or $\mathsf{\Pi}_0$.

The value of dual-mode proof systems emerges when the common reference strings created in the two modes are indistinguishable. To formalize this, consider game $\mathbf{G}^{\mathrm{mode}}_{\mathsf{D\Pi}}$ of Figure 11 associated to dual-mode proof system $\mathsf{D\Pi}$, and let the mode advantage of adversary A be defined by $\mathbf{Adv}^{\mathrm{mode}}_{\mathsf{D\Pi}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{mode}}_{\mathsf{D\Pi}}(\mathrm{A})] - 1$.

Theorem 4.8 says that if one of the induced proof systems $\mathsf{\Pi}_\mu$ satisfies ZK with respect to some simulator S, and the modes are indistinguishable, then the other induced proof system $\mathsf{\Pi}_{1-\mu}$ also satisfies ZK with respect to the same S. The statement is in the concrete security framework. Given an adversary A_{zk} against the ZK of $\mathsf{\Pi}_{1-\mu}$, the reduction can be viewed as constructing two adversaries, one against the ZK of $\mathsf{\Pi}_\mu$ and the other, called $\mathrm{A}_{\mathrm{mode}}$, against the mode indistinguishability of $\mathsf{D\Pi}$, but the first construction is trivial, returning A_{zk} itself.

Theorem 4.8 *Let $\mathsf{D\Pi}$ be a dual-mode proof system for claim validator CV, and let $\mathsf{\Pi}_1, \mathsf{\Pi}_0$ be the induced proof systems. Let $\mu \in \{0,1\}$. Given an adversary A_{zk}, we can construct an adversary $\mathrm{A}_{\mathrm{mode}}$ (shown explicitly in Figure 11), such that*

$$\mathbf{Adv}^{\mathrm{zk}}_{\mathsf{\Pi}_{1-\mu},\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) \leq \mathbf{Adv}^{\mathrm{zk}}_{\mathsf{\Pi}_{\mu},\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) + \mathbf{Adv}^{\mathrm{mode}}_{\mathsf{D\Pi}}(\mathrm{A}_{\mathrm{mode}}) .$$

Adversary $\mathrm{A}_{\mathrm{mode}}$ has about the same running time as adversary A_{zk}.

Exercise 4.9 *Prove Theorem 4.8.*

The literature suggests that a claim analogous to Theorem 4.8 is also true for soundness, meaning if Π_μ is SND1 and $\mathsf{D\Pi}$ is mode indistinguishable, then $\Pi_{1-\mu}$ is also SND1. (In particular such a claim underlies the claim of GOS [GOS06] that their proof system is perfect zero-knowledge and computationally sound.) But I don't see a way to establish such a claim, at least in general and for the definition of SND1 we are using. The difficulty is that testing whether or not SND1 is violated involves testing membership in the language of true claims, and this in general (and in cases of interest) is hard (in asymptotic language, not polynomial time.) Does this indicate a gap in GOS [GOS06]? Hard to say, since their definition of soundness is somewhat ambiguous.

The definitional issue can be understood by considering two ways to formalize SND1, namely in the penalty style (call this SND1-P) and the exclusion style (call it SND1-E). These two styles, and their issues, were first formally considered in [BHK15] in the context of IND-CCA public-key encryption and KEMs. In the penalty style, which is the one we have used (meaning, SND1-P is our SND1) the adversary is penalized by the game when it submits a verification query where the claim is outside the language, the game testing this and not allowing it to win in this case. In the exclusion style, the adversary is simply prohibited from making queries with claims outside the language, meaning a claim of SND1-E security refers only to the sub-class of adversaries that never make such queries. For SND1-E, we should be able to show the transference under mode indistinguishability, meaning Π_μ is SND1-E and $\mathsf{D\Pi}$ is mode indistinguishable implies $\Pi_{1-\mu}$ is also SND1-E. But for SND1-P, the claim seems likely to be false in general.

Exercise 4.10 *Develop results analogous to Theorem 4.8 for as many of the other notions as possible.*

We will see later how the dual-mode perspective underlies proof system constructions in the literature and enables us to both better understand them and to state stronger results about them.

4.7 Relations

The following says that for SND1, security against adversaries making one VF query implies it against adversaries making q queries, up to a factor q increase in advantage. Asymptotically, this means that SND1 can restrict attention to single-query adversaries. Concretely, in applications requiring security against many-query adversaries, we can factor in the degradation if starting from security results about single-query adversaries. More interestingly, it opens the door to tighter (better) reductions for particular proof systems that do not lose this factor of q, allowing efficiency gains through instantiation with smaller security parameters.

Proposition 4.11 *Let* TC *be a true-claim language and* Π *a proof system. Let* A *be an adversary making* q *queries to its* VF *oracle. Then we can construct an adversary* A_1 *(shown explicitly in Figure 12), making only one query to its* VF *oracle, such that*

$$\mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}) \leq q \cdot \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}_1) \, .$$

The running time of A_1 *is about that of* A.

The proof would appear to be standard. Namely, let A_1 make a random guess $g \leftarrow\!\!\$\, [1..q]$, forward the g-th VF query of A to its own oracle, and answer the other queries on its own. This, however, runs into a difficulty, namely that that at line 1 of Figure 5, oracle VF tests membership of x in $\mathsf{TC}(\mathrm{crs})$, and A_1 cannot do this test, since it may not be (and for proof systems of interest, Games $\boxed{\mathrm{G}_0}$, G_1

$\textsc{Init}()$:

1 $\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$; Return crs

$\textsc{Vf}(x, \mathrm{pf})$:

2 $i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$; $\mathrm{vf}_i \leftarrow \mathsf{false}$

3 If ($\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$ and $(x \notin \mathsf{TC}(\mathrm{crs}))$) then

4 $\quad \mathsf{win} \leftarrow \mathsf{true}$; $\boxed{\mathrm{vf}_i \leftarrow \mathsf{true}}$

5 Return vf_i

$\textsc{Fin}()$:

6 Return win

Game $\mathrm{G}_{2,\ell}$

$\textsc{Init}()$:

1 $\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$; Return crs

$\textsc{Vf}(x, \mathrm{pf})$:

2 $i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$

3 Return false

$\textsc{Fin}()$:

4 Return ($\mathsf{\Pi.V}(\mathrm{crs}, x_\ell, \mathrm{pf}_\ell)$ and $(x_\ell \notin \mathsf{TC}(\mathrm{crs}))$)

Adversary A_1:

1 $\mathrm{crs} \leftarrow \mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TC}}.\textsc{Init}$; $\mathrm{A}^{\textsc{Init},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}()$:

2 Return crs

$\textsc{Vf}(x, w)$:

3 $i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$

4 Return false

$\textsc{Fin}()$:

5 $g \leftarrow\!\!\$\ [1..q]$

6 $\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TC}}.\textsc{Vf}(x_g, \mathrm{pf}_g)$

7 $\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TC}}.\textsc{Fin}$

Figure 12: Games and adversary for Proposition 4.11.

isn't) efficiently doable. (In asymptotic language, it is not polynomial time.) The proof below gets around this before doing the guessing.

Proof of Proposition 4.11: Games $\mathrm{G}_0, \mathrm{G}_1$ of Figure 12 index the adversary's oracle queries for future reference. Game G_0 includes the boxed code. As a consequence we have

$$\mathbf{Adv}^{\mathrm{snd1}}_{\mathsf{\Pi},\mathsf{TC}}(\mathrm{A}) = \Pr[\mathrm{G}_0(\mathrm{A})] \ .$$

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-win. Lemma 3.1 thus implies that they set win with the same probability. However, win is also what they return, and once win is set to true it is never set back to false, so the games return true with the same probability, meaning

$$\Pr[\mathrm{G}_0(\mathrm{A})] = \Pr[\mathrm{G}_1(\mathrm{A})] \ .$$

In game G_1, the line 5 reply to any $\textsc{Vf}$ query is false. This is what the game-playing has accomplished, namely it has allowed us to move to a game where we can reply to oracle queries in a trivial way and in particular without needing to test membership in TC.

Assume wlog that A always makes exactly q oracle queries. (Not fewer.) Games $\mathrm{G}_{2,\ell}$ of Figure 12 are defined for all $\ell \in [1..q]$. The line 3 test in game G_1 is pushed, in these games, to $\textsc{Fin}$, with an

Games $\boxed{\mathrm{G}_0}$, G_1

$\textsc{Init}()$:

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\; \mathsf{S.C}$; Return crs

$\textsc{Ex}(x, \mathrm{pf})$:

If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false

$i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$; $\mathrm{vf}_i \leftarrow \mathsf{true}$

$w \leftarrow\!\!\$\; \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, x, \mathrm{pf})$

If $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$ then

$\quad \mathsf{win} \leftarrow \mathsf{true}$; $\boxed{\mathrm{vf}_i \leftarrow \mathsf{false}}$

Return vf_i

$\textsc{Fin}()$:

Return win

Game $\mathrm{G}_{2,\ell}$

$\textsc{Init}()$:

$\mathrm{crs} \leftarrow\!\!\$\; \mathsf{\Pi.C}$; Return crs

$\textsc{Ex}(x, \mathrm{pf})$:

If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false

$i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$

Return true

$\textsc{Fin}()$:

$w \leftarrow\!\!\$\; \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, x_\ell, \mathrm{pf}_\ell)$

Return $(\mathsf{CV}(\mathrm{crs}, x_\ell, w) = \mathsf{false})$

Adversary A_1:

$\mathrm{crs} \leftarrow \mathbf{G}^{\mathrm{xt1}}_{\mathsf{\Pi,CV}}.\textsc{Init}$; $\mathrm{A}^{\textsc{Init},\textsc{Ex},\textsc{Fin}}$

$\textsc{Init}()$:

Return crs

$\textsc{Ex}(x, w)$:

If $(\mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf}) = \mathsf{false})$ then return false

$i \leftarrow i+1$; $x_i \leftarrow x$; $\mathrm{pf}_i \leftarrow \mathrm{pf}$

Return true

$\textsc{Fin}()$:

$g \leftarrow\!\!\$\; [1..q]$

$\mathbf{G}^{\mathrm{xt1}}_{\mathsf{\Pi,CV}}.\textsc{Ex}(x_g, \mathrm{pf}_g)$

$\mathbf{G}^{\mathrm{xt1}}_{\mathsf{\Pi,CV}}.\textsc{Fin}$

Figure 13: Games and adversary for Proposition 4.12.

additional twist, namely that it is performed on only one of the queries, namely the ℓ-th. Then

$$\begin{aligned}\mathbf{Adv}^{\mathrm{snd1}}_{\mathsf{\Pi,TC}}(\mathrm{A}_1) &= \frac{1}{q} \cdot \sum_{i=1}^{q} \Pr[\mathrm{G}_{2,i}(\mathrm{A})] \\ &\geq \frac{1}{q} \cdot \Pr[\mathrm{G}_1(\mathrm{A})] \ ,\end{aligned}$$

where the last inequality is by the union bound. ∎

Proposition 4.12 *Let* CV *be a claim validator and* $\mathsf{\Pi}$ *a proof system. Let* A *be an adversary making* q *queries to its* $\textsc{Ex}$ *oracle. Then we can construct an adversary* A_1 *(shown explicitly in Figure 13), making only one query to its* $\textsc{Ex}$ *oracle, such that*

$$\mathbf{Adv}^{\mathrm{xt1}}_{\mathsf{\Pi,CV}}(\mathrm{A}) \leq q \cdot \mathbf{Adv}^{\mathrm{xt1}}_{\mathsf{\Pi,CV}}(\mathrm{A}_1) \ .$$

The running time of A_1 *is about that of* A.

Proof of Proposition 4.12: Games $\mathrm{G}_0, \mathrm{G}_1$ of Figure 13 index the adversary's oracle queries for future reference. Game G_0 includes the boxed code. As a consequence we have

$$\mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV}}(\mathrm{A}) = \Pr[\mathrm{G}_0(\mathrm{A})] \ .$$

Games $\mathrm{G}_0, \mathrm{G}_1$ are identical-until-win. Lemma 3.1 thus implies that they set win with the same probability. However, win is also what they return, and once win is set to true it is never set back to false, so the games return true with the same probability, meaning

$$\Pr[\mathrm{G}_0(\mathrm{A})] = \Pr[\mathrm{G}_1(\mathrm{A})] \ .$$

In game G_1, the line 5 reply to any $\textsc{Vf}$ query is true. This is what the game-playing has accomplished, namely it has allowed us to move to a game where we can reply to oracle queries in a trivial way and in particular without needing to test whether a witness is valid, which an adversary cannot do since it does not have the witness.

Assume wlog that A always makes exactly q oracle queries. (Not fewer.) Games $\mathrm{G}_{2,\ell}$ of Figure 13 are defined for all $\ell \in [1..q]$. The line 5 test in game G_1 is pushed, in these games, to $\textsc{Fin}$, with an additional twist, namely that it is performed on only one of the queries, namely the ℓ-th. Then

$$\begin{aligned}\mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV}}(\mathrm{A}_1) &= \frac{1}{q} \cdot \sum_{i=1}^{q} \Pr[\mathrm{G}_{2,i}(\mathrm{A})] \\ &\geq \frac{1}{q} \cdot \Pr[\mathrm{G}_1(\mathrm{A})] \ ,\end{aligned}$$

where the last inequality is by the union bound. ∎

5 Groups and bilinear maps

If $\mathbb{G}$ is a group, then its identity element is denoted $\mathbb{1}_{\mathbb{G}}$. If $h \in \mathbb{G}$ then $\langle h \rangle_{\mathbb{G}} = \{\, h^i : i \in \mathbb{Z} \,\}$ is the cyclic subgroup of $\mathbb{G}$ generated by h. We let $\mathrm{ord}_{\mathbb{G}}(h) = |\langle h \rangle_{\mathbb{G}}| \geq 1$ be the order of h in $\mathbb{G}$, which is also the smallest positive integer m such that $h^m = \mathbb{1}_{\mathbb{G}}$. Recall that $\mathrm{ord}_{\mathbb{G}}(h)$ divides the order $n = |\mathbb{G}|$ of the group for all $h \in \mathbb{G}$. We say that $g \in \mathbb{G}$ is a generator of $\mathbb{G}$ if $\langle g \rangle_{\mathbb{G}} = \mathbb{G}$, and denote by $\mathrm{Gen}(\mathbb{G})$ the set of all generators of $\mathbb{G}$. The group $\mathbb{G}$ is cyclic if $\mathrm{Gen}(\mathbb{G}) \neq \emptyset$. If $h \in \mathbb{G}$ has order m, then the discrete logarithm function $\mathrm{dlog}_{\mathbb{G},h} \colon \mathbb{G} \to \mathbb{Z}_m \cup \{\bot\}$, on input $A \in \mathbb{G}$, returns the unique $i \in \mathbb{Z}_m$ such that $A = g^i$ if $A \in \langle h \rangle_{\mathbb{G}}$, and otherwise returns $\bot$.

The following is sometimes called the fundamental theorem of cyclic groups.

Proposition 5.1 *Let $\mathbb{G}$ be any cyclic group of order n. Let $g \in \mathrm{Gen}(\mathbb{G})$ and let m be any divisor of n. Then:*

1. *Every subgroup of $\mathbb{G}$ is cyclic.*

2. *$\mathbb{G}$ has exactly one subgroup of order m, and $g^{n/m}$ generates it.*

If m is a divisor of the order of cyclic group $\mathbb{G}$, then we will denote by $\mathbb{G}_m$ the unique (cyclic) subgroup of $\mathbb{G}$ of order m.

Exercise 5.2 *Prove Proposition 5.1.*

Game $\mathbf{G}^{\mathrm{subd}}_{\mathsf{GG}}$

INIT():

$(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\!\!\$\ \mathsf{GG}$; $b \leftarrow\!\!\$\ \{0,1\}$; $g \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$

If $(b = 1)$ then $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G}_q)$ else $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$

Return $(n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h$

FIN(b'):

Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{dlin}}_{\mathsf{GG}}$

INIT():

$(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\!\!\$\ \mathsf{GG}$; $b \leftarrow\!\!\$\ \{0,1\}$; $g \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$

$x, y, z, r, s \leftarrow\!\!\$\ \mathbb{Z}_n$

If $(b = 1)$ then $Z \leftarrow g^{r+s}$ else $Z \leftarrow g^{z}$

Return $(n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g^x, g^y, g^{rx}, g^{sy}, Z$

FIN(b'):

Return $(b' = b)$

Figure 14: Games defining SUBD (Subgroup Decision) and DLIN (Decision Linear) problems relative to group generator GG.

Let $\mathbb{G}, \mathbb{T}$ be cyclic groups, both of the same order n. We refer to $\mathbb{T}$ as the target group. The order n of the groups may be prime or composite. A bilinear map (also called a pairing) is a function $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ with the following properties:

(1) Bilinearity: $\mathbf{e}(A, B_1 B_2) = \mathbf{e}(A, B_1) \cdot \mathbf{e}(A, B_2)$ for all $A, B_1, B_2 \in \mathbb{G}$, and $\mathbf{e}(A_1 A_2, B) = \mathbf{e}(A_1, B) \cdot \mathbf{e}(A_2, B)$ for all $A_1, A_2, B \in \mathbb{G}$.

(2) Non-degeneracy: If $g \in \mathrm{Gen}(\mathbb{G})$ then $\mathbf{e}(g, g) \in \mathrm{Gen}(\mathbb{T})$.

Proposition 5.3 *Let $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ be a bilinear map, and let $n = |\mathbb{G}| = |\mathbb{T}|$. Then*

1. $\mathbf{e}(g_1^{x_1}, g_2^{x_2}) = \mathbf{e}(g_1, g_2)^{x_1 x_2}$ *for all $g_1, g_2 \in \mathbb{G}$ and all $x_1, x_2 \in \mathbb{Z}$.*

2. $\mathbf{e}(A, B) = \mathbf{e}(B, A)$ *for all $A, B \in \mathbb{G}$.*

3. $\mathbf{e}(A, \mathbb{1}_{\mathbb{G}}) = \mathbf{e}(\mathbb{1}_{\mathbb{G}}, A) = \mathbb{1}_{\mathbb{T}}$ *for all $A \in \mathbb{G}$.*

4. *Let $g \in \mathbb{G}$. Define $f_g\colon \mathbb{G} \to \mathbb{T}$ by $f_g(A) = \mathbf{e}(g, A)$. Then f_g is a group homomorphism, and if $g \in \mathrm{Gen}(\mathbb{G})$, it is an isomorphism. Likewise for $f_g\colon \mathbb{G} \to \mathbb{T}$ defined by $f_g(A) = \mathbf{e}(A, g)$.*

Exercise 5.4 *Prove Proposition 5.3.*

We will rely on the conjectured computational hardness of various problems about groups equipped with a bilinear map. The historically first of these problems was the Computational Bilinear Diffie Hellman (CBDH) problem [BF03]. The game picks $g \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$ and $x, y, z \leftarrow\!\!\$\ \mathbb{Z}_n$, where $n = |\mathbb{G}|$, and gives g, g^x, g^y, g^z to the adversary. To win, the adversary must return $\mathbf{e}(g, g)^{xyz}$. Its cbdh-advantage is the probability that it wins.

Of interest to us is a different problem, called the Subgroup Decision (SUBD) problem [BGN05]. Here, the order $n = pq$ of the (cyclic) groups $\mathbb{G}, \mathbb{T}$ is a composite number that (like an RSA modulus) is a product of distinct, odd primes p, q. The game picks $g \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$ and a challenge bit $b \leftarrow \{0, 1\}$. If $b = 1$, it lets $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G}_q)$ be a generator of the unique order-q subgroup of $\mathbb{G}$, and otherwise it lets $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$ be a generator of the full group $\mathbb{G}$. It gives g, h to the adversary, asking the latter to provide a guess $b' \in \{0, 1\}$ of the value of b, and the adversary wins if $b' = b$. The subd-advantage of the adversary is $2w - 1$, where w is the probability that it wins.

If the adversary is given q, it is easy for it to win the above game. (Why?) So in the formalization, the adversary is not given q, meaning not given the factorization of the group order n. This leads us to ask how exactly the problem is formalized.

In defining the CBDH problem, we regarded the groups $\mathbb{G}, \mathbb{T}$, their order n and the bilinear map $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ as fixed. This works for many problems, but not all, and in particular not for SUBD, because if $n = pq$ is fixed it is not clear in what sense q can be viewed as hidden or unknown to the adversary. Instead, we draw the groups from some distribution. Doing this is the responsibility of a *group generator* GG, and it parameterizes the problem, so that we have SUBD not by itself, but with respect to GG.

As an analogy, when we define the Discrete Logarithm (DL) problem in a concrete security setting, we can do so with respect to a fixed (cyclic) group and generator. But when we define the RSA problem, it is with respect to an RSA generator.

In the literature, we typically see the setup written something like $(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\$\ \mathsf{GG}$. We will also use such notation, but it is worth understanding all that it shoves under the rug. The notation seems to say that the generator returns the groups and bilinear map, but this of course is not possible, for these objects are too big to be efficiently specified. What GG actually returns is some short description string η which implicitly specifies the groups and map through algorithms, associated to GG, that allow some set of desired operations to be performed. There is an algorithm for testing membership in the first group; given η and a description "A" of a group element, it returns true if A is in the group and false otherwise. The group $\mathbb{G}$ is now implicitly defined as the set of all A such that this algorithm returns true given η and "A." There is an algorithm to perform the group operation on this group, taking η and the descriptions "A", "B" of two group elements to return the description of the group element AB. There are analogous algorithms for the second group $\mathbb{T}$. There are algorithms for computing inverses in the groups. There is an algorithm that defines $\mathbf{e}$ by computing it. There may be algorithms for sampling random elements or generators of these groups. And so on.

If we were to proceed in strict formality, we would make these algorithms explicit and invoke them in our games, but this would be rather tedious and hard to follow. The compromise is an informal setup step written $(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\$\ \mathsf{GG}$, followed by usage of the groups with the usual mathematical operations.

Our convention on group generators is that $(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\$\ \mathsf{GG}$ means the groups are cyclic of order n, the map $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ is bilinear and q is a prime dividing n. We may then distinguish a few special classes of group generators. We say that GG is a prime-order group generator if $n = q$. We say it is a 2-prime group generator if q and n/q are distinct, odd primes. This allows us to capture most generators of interest in a common notation. Most definitions apply to all classes of generators, but the hardness of the problem may depend on the choice of generator.

With these conventions, the game formalizing the SUBD problem relative to group generator GG is on the left in Figure 14, and the subd-advantage of adversary A is defined by $\mathbf{Adv}^{\mathrm{subd}}_{\mathsf{GG}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{subd}}_{\mathsf{GG}}(\mathrm{A})] - 1$. In usage, GG will be a 2-prime group generator.

On the right in Figure 14 is a game formalizing the Decision Linear (DLIN) problem [BBS04] relative to group generator GG. The dlin-advantage of adversary A is defined by $\mathbf{Adv}^{\mathrm{dlin}}_{\mathsf{GG}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{dlin}}_{\mathsf{GG}}(\mathrm{A})] - 1$. Here GG could be either a prime-order group generator or a 2-prime group generator.

Algorithm $\mathsf{DCS.P}(\mu)$:	Algorithm $\mathsf{DCS.C}(cp, m, d)$:
$(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\$ \mathsf{GG}$	$(n, \mathbb{G}, g, h) \leftarrow cp$
$g \leftarrow\$ \mathrm{Gen}(\mathbb{G})$	$c \leftarrow g^m h^d$
If $(\mu = 1)$ then $h \leftarrow\$ \mathrm{Gen}(\mathbb{G}_q)$ else $h \leftarrow\$ \mathrm{Gen}(\mathbb{G})$	Return c
$cp \leftarrow (n, \mathbb{G}, g, h)$; Return cp	

Figure 15: Algorithms of the dual-mode commitment scheme DCS for Exercise 6.2.

6 Commitment from SUBD

As a warmup, we discuss the BGN method for commitment or encryption [BGN05]. Suppose $n = pq$ is a product of distinct, odd primes p, q, and $\mathbb{G}$ is a cyclic group of order n. Proposition 5.1 says that $\mathbb{G}$ has a unique subgroup of order q that we denote $\mathbb{G}_q$, and this subgroup is itself cyclic. For $g \in \mathrm{Gen}(\mathbb{G})$ and $h \in \mathrm{Gen}(\mathbb{G}_q)$, define $\mathrm{E}_{g,h} \colon \mathbb{Z}_n \times \mathbb{Z} \to \mathbb{G}$ by $\mathrm{E}_{g,h}(m, r) = g^m h^r$. We think of this function as encrypting message $m \in \mathbb{Z}_n$ under coins r to produce ciphertext $c \leftarrow \mathrm{E}_{g,h}(m, r)$, or, alternatively, as committing to m with randomness r to produce commitment c, in the style of Pedersen [Ped92]. For $m \in \mathbb{Z}_n$ let $\mathrm{E}_{g,h}(m) = \{ \mathrm{E}_{g,h}(m, r) \,:\, r \in \mathbb{Z} \}$ be the set of ciphertexts corresponding to message m.

The encryption is not always uniquely reversible. However, part **1** of Exercise 6.1 says that given $c \in \mathrm{E}_{g,h}(m)$ for some $m \in \mathbb{Z}_p$, one can (in principle) uniquely recover m, making the encryption reversible for messages in $\mathbb{Z}_p$. But this decryption operation may not be efficiently computable. It becomes so if we know that m is drawn from some small set $M \subseteq \mathbb{Z}_p$. List the elements of M as $m_1 < m_2 < \ldots < m_{|M|}$, and define:

Algorithm $D_{M,g,h}(c)$

For $j = 1, \ldots, |M|$ do

  If ($(g^{-m_j} c)^q = \mathbb{1}_{\mathbb{G}}$) then return m_j.

Then part **2** of Exercise 6.1 says that given input $c \in \mathrm{E}_{g,h}(m^*)$ for $m^* \in M \subseteq \mathbb{Z}_p$, the above procedure returns m^*.

Exercise 6.1 *Let $n = pq$ where p, q are distinct, odd primes. Let $\mathbb{G}$ be a cyclic group of order n. Let $g \in \mathrm{Gen}(\mathbb{G})$ and let $h \in \mathrm{Gen}(\mathbb{G}_q)$. Let $M \subseteq \mathbb{Z}_p$. Prove that:*

1. *If $m_1, m_2 \in \mathbb{Z}_p$ are distinct then $\mathrm{E}_{g,h}(m_1) \cap \mathrm{E}_{g,h}(m_2) = \emptyset$*

2. *If $m^* \in M$ and $c \in \mathrm{E}_{g,h}(m^*)$ then $D_{M,g,h}(c) = m^*$.*

Exercise 6.2 *Let $\ell \geq 1$ be an integer and GG a 2-prime group generator such that $\ell < \min(q, n/q)$ for all $(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \in [\mathsf{GG}]$. Let DCS be the dual-mode commitment scheme whose algorithms are shown in Figure 15, and where we set $\mathsf{DCS.M}((n, \mathbb{G}, g, h)) = \mathbb{Z}_\ell$ and $\mathsf{DCS.D}((n, \mathbb{G}, g, h)) = \mathbb{Z}_n$. (1) Prove that DCS_1 has perfect binding (2) Prove that DCS_0 has perfect hiding (3) Formulate and prove a result saying that DCS is mode-indistinguishable assuming SUBD for GG. (4) Draw conclusions about the hiding security of DCS_1 and the binding security of DCS_0.*

7 NIZKs from SUBD

In this section we consider the Groth, Ostrovsky and Sahai (GOS) [GOS06] NIZK that is based on the assumed hardness of the SUBD problem. We cast it as a dual-mode proof system, something that is implicit but not explicit in [GOS06]. The system makes use of the commitment or encryption method discussed in Section 6.

In the GOS NIZK, we will encrypt messages m that are either 0 or 1. The system begins with a protocol to show that a given ciphertext c is an encryption of either 0 or 1, in zero-knowledge, meaning, in particular, without revealing whether the decryption is 0 or it is 1. We will call this the Zero-or-One system. We start with an algebraic characterization of the property of encrypting either 0 or 1. This Lemma does not appear explicitly in [GOS06]; we have extracted it from claims present in the text and worked out a detailed proof.

Lemma 7.1 *Let $n = pq$ where p, q are distinct, odd primes. Let $\mathbb{G}, \mathbb{T}$ be cyclic groups of order n, and let $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{T}$ be a bilinear map. Let $g \in \mathrm{Gen}(\mathbb{G})$ and let $h \in \mathrm{Gen}(\mathbb{G}_q)$. For $m \in \mathbb{Z}_n$ let $\mathrm{E}_{g,h}(m) = \{\, g^m h^r : r \in \mathbb{Z} \,\}$. Let $c \in \mathbb{G}$. Then the following are equivalent:*

1. $c \in \mathrm{E}_{g,h}(0) \cup \mathrm{E}_{g,h}(1)$

2. $\{c, cg^{-1}\} \cap \mathbb{G}_q \neq \emptyset$

3. $\mathbf{e}(c, cg^{-1})^q = \mathbb{1}_{\mathbb{T}}$.

Proof of Lemma 7.1: Suppose **1** holds. So $c = g^m hr^r$ for some $m \in \{0, 1\}$ and some $r \in \mathbb{Z}$. If $m = 0$ then $c^q = g^{mq}h^{rq} = g^{mq} = \mathbb{1}_{\mathbb{G}}$, so $c \in \mathbb{G}_q$. If $m = 1$ then $(cg^{-1})^q = g^{mq-q}h^{rq} = g^{q(m-1)} = \mathbb{1}_{\mathbb{G}}$, so $cg^{-1} \in \mathbb{G}_q$. So **2** holds.

Now assume **2** holds. If, on the one hand, $c \in \mathbb{G}_q$ then Proposition 5.3 implies that $\mathbf{e}(c, cg^{-1})^q = \mathbf{e}(c^q, cg^{-1}) = \mathbf{e}(\mathbb{1}_{\mathbb{G}}, cg^{-1}) = \mathbb{1}_{\mathbb{T}}$. If, on the other hand, $cg^{-1} \in \mathbb{G}_q$ then $\mathbf{e}(c, cg^{-1})^q = \mathbf{e}(c, (cg^{-1})^q)$ $= \mathbf{e}(c, \mathbb{1}_{\mathbb{G}}) = \mathbb{1}_{\mathbb{T}}$. So **3** holds.

Now assume **3** holds. Since $g \in \mathrm{Gen}(\mathbb{G})$, there is a $k \in [0..n-1]$ such that $c = g^k$. Then $\mathbb{1}_{\mathbb{T}} = \mathbf{e}(c, cg^{-1})^q = \mathbf{e}(g^k, g^{k-1})^q = \mathbf{e}(g, g)^{k(k-1)q}$. Since $\mathbf{e}(g, g) \in \mathrm{Gen}(\mathbb{T})$, it must be that $n \mid k(k-1)q$, which means $p \mid k(k-1)$. Since p is prime, either $p \mid k$ or $p \mid (k-1)$. If $p \mid k$ then $n \mid kq$ so $c^q = g^{kq} = \mathbb{1}_{\mathbb{G}}$. If $p \mid (k-1)$ then $n \mid (k-1)q$ so $(cg^{-1})^q = g^{(k-1)q} = \mathbb{1}_{\mathbb{G}}$. So **2** holds.

Now assume **2** holds. Suppose $c \in \mathbb{G}_q$. Then $c = h^r$ for some $r \in [0..q-1]$, so $c = g^0 h^r$, putting c in $\mathrm{E}_{g,h}(0)$. Now suppose $cg^{-1} \in \mathbb{G}_q$. Then $cg^{-1} = h^r$ for some $r \in [0..q-1]$, so $c = g^1 h^r$, putting c in $\mathrm{E}_{g,h}(1)$. So **1** holds. ▮

Assume the prover and verifier know c. The prover wants to show that item **1** of Lemma 7.1 is true. Item **3** suggests that the verifier can simply check that $\mathbf{e}(c, cg^{-1})^q = \mathbb{1}_{\mathbb{T}}$. But this requires the verifier to know q, and is then not ZK, because given q one can, via the above procedure, decrypt $c \in \mathrm{E}_{g,h}(0) \cup \mathrm{E}_{g,h}(1)$ to obtain the underlying message $m = \mathrm{E}_{g,h}^{-1}(c)$. Instead, we consider the dual-model proof system $\mathsf{D\Pi}^{0\text{or}1}$ (we call it the Zero-or-One system), and corresponding claim validator $\mathsf{CV}^{0\text{or}1}$, that are described via Figure 17.

Theorem 7.2 says that, assuming SUBD, the modes are indistinguishable.

Adversary $\mathrm{A}_{\mathrm{subd}}$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow\!\$\; \mathbf{G}^{\mathrm{subd}}_{\mathsf{GG}}.\textsc{Init}$

$\mathrm{A}^{\textsc{Init},\textsc{Fin}}_{\mathrm{mode}}$

$\textsc{Init}$:

Return $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h)$

$\textsc{Fin}(b')$:

$\mathbf{G}^{\mathrm{subd}}_{\mathsf{GG}}.\textsc{Fin}(b')$

Figure 16: Adversary for Theorem 7.2.

Theorem 7.2 *Let* GG *be a group generator. Let* $\mathsf{D\Pi}^{0\mathrm{or}1}$ *be the associated zero-or-one dual-mode proof system as per the left of Figure 17. Given an adversary* $\mathrm{A}_{\mathrm{mode}}$*, we can construct an adversary* $\mathrm{A}_{\mathrm{subd}}$ *(shown explicitly in Figure 16), such that*

$$\mathbf{Adv}^{\mathrm{mode}}_{\mathsf{D\Pi}^{0\mathrm{or}1}}(\mathrm{A}_{\mathrm{mode}}) \leq \mathbf{Adv}^{\mathrm{subd}}_{\mathsf{GG}}(\mathrm{A}_{\mathrm{subd}}) \ .$$

Adversary $\mathrm{A}_{\mathrm{subd}}$ *has about the same running time as adversary* $\mathrm{A}_{\mathrm{mode}}$.

Exercise 7.3 *Prove Theorem 7.2.*

Theorem 7.4 makes a few claims about this system. Recall that a dual-mode proof system $\mathsf{D\Pi}^{0\mathrm{or}1}$ induces two, standard proof systems Π_0, Π_1. The theorem says that Π_1 has perfect soundness, and Π_0 has perfect zero knowledge, both relative to $\mathsf{CV}^{0\mathrm{or}1}$.

Claim validator $\mathsf{CV}^{0\mathrm{or}1\text{-}\mathrm{msg}}$, shown on the right in Figure 17, defines the same language as $\mathsf{CV}^{0\mathrm{or}1}$, meaning $\mathsf{TrCl}_{\mathsf{CV}^{0\mathrm{or}1}}(\mathrm{crs}) = \mathsf{TrCl}_{\mathsf{CV}^{0\mathrm{or}1\text{-}\mathrm{msg}}}(\mathrm{crs})$ for all crs, the difference being that the witness for $\mathsf{CV}^{0\mathrm{or}1}$ is both message and randomness, while for $\mathsf{CV}^{0\mathrm{or}1\text{-}\mathrm{msg}}$ it is only the message. This is not defined in GOS [GOS06], and we'll get to it in Theorem 7.5.

Theorem 7.4 *Let* GG *be a group generator. Let* $\mathsf{D\Pi}^{0\mathrm{or}1}$ *be the associated zero-or-one dual-mode proof system, and* $\mathsf{CV}^{0\mathrm{or}1}$ *the associated claim validator, as per the left of Figure 17. Let* $\mathsf{TC} = \mathsf{TrCl}_{\mathsf{CV}^{0\mathrm{or}1}}$ *be the true-claim language for this claim validator. Let* Π_1, Π_0 *be the proof systems induced by* $\mathsf{D\Pi}^{0\mathrm{or}1}$*. Then:*

1. Π_1 *and* Π_0 *satisfy perfect completeness for* $\mathsf{CV}^{0\mathrm{or}1}$

2. Π_1 *satisfies perfect soundness for* TC*:* $\mathbf{Adv}^{\mathrm{snd1}}_{\Pi_1,\mathsf{TC}}(\mathrm{A}) = 0$ *for all adversaries* A.

3. Π_0 *satisfies perfect zero knowledge for* $\mathsf{CV}^{0\mathrm{or}1}$ *relative to the simulator* $\mathsf{S}^{0\mathrm{or}1}$ *shown on the right of Figure 17:* $\mathbf{Adv}^{\mathrm{zk}}_{\Pi_0,\mathsf{CV}^{0\mathrm{or}1},\mathsf{S}^{0\mathrm{or}1}}(\mathrm{A}) = 0$ *for all adversaries* A.

Theorem 7.4 states properties of two proof systems, Π_1 and Π_0. In usage, of course, one must make a choice of which of the two proof systems to deploy or use. If we take Π_1 then we have perfect completeness and soundness and, by combining Theorems 7.4, 7.2 and 4.8, computational zero knowledge. A concrete-security statement for the zero knowledge can be obtained from the stated theorems.

However, the proof system of greater interest to GOS [GOS06] (as per the title of the paper) is Π_0 (meaning, mode 0), because it is perfect zero-knowledge. For this system, the paper also claims

Claim validator $\mathsf{CV}^{0\text{or}1}(\text{crs}, c, w)$:

1 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$; $(m, r) \leftarrow w$
2 If $(m \notin \{0, 1\})$ then return false
3 Return $(c = g^m h^r)$

CRS generator $\mathsf{D\Pi}^{0\text{or}1}.\mathsf{C}(\mu)$: // $\mu \in \{0, 1\}$

4 $(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\!\$\ \mathsf{GG}$
5 If $(\mu = 1)$ then $g \leftarrow\!\$\ \text{Gen}(\mathbb{G})$; $h \leftarrow\!\$\ \text{Gen}(\mathbb{G}_q)$; $\text{td} \leftarrow q$
6 Else $h \leftarrow\!\$\ \text{Gen}(\mathbb{G})$; $\gamma \leftarrow\!\$\ \mathbb{Z}_n^*$; $g \leftarrow h^\gamma$; $\text{td} \leftarrow (q, \gamma)$
7 $\text{crs} \leftarrow ((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h)$; Return (crs, td)

Proof generator $\mathsf{D\Pi}^{0\text{or}1}.\mathsf{P}(\text{crs}, c, w)$:

8 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$
9 $(m, r) \leftarrow w$
10 $s \leftarrow\!\$\ \mathbb{Z}_n^*$; $t \leftarrow s^{-1} \bmod n$; $u \leftarrow g^{2m-1} h^r$
11 $\text{pf}_1 \leftarrow h^s$; $\text{pf}_2 \leftarrow u^{rt}$; $\text{pf}_3 \leftarrow g^s$
12 $\text{pf} \leftarrow (\text{pf}_1, \text{pf}_2, \text{pf}_3)$; Return pf

Proof verifier $\mathsf{D\Pi}^{0\text{or}1}.\mathsf{V}(\text{crs}, c, \text{pf})$:

13 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$
14 If $(c \notin \mathbb{G}$ or $\text{pf} \notin \mathbb{G}^3)$ then return false
15 $(\text{pf}_1, \text{pf}_2, \text{pf}_3) \leftarrow \text{pf}$
16 $\text{vf}_1 \leftarrow (\mathbf{e}(c, cg^{-1}) = \mathbf{e}(\text{pf}_1, \text{pf}_2))$
17 $\text{vf}_2 \leftarrow (\mathbf{e}(\text{pf}_1, g) = \mathbf{e}(h, \text{pf}_3))$
18 Return $(\text{vf}_1 \wedge \text{vf}_2)$

Sim CRS generator $\mathsf{S}^{0\text{or}1}.\mathsf{C}$:

19 $(\text{crs}, \text{td}) \leftarrow\!\$\ \mathsf{D\Pi}^{0\text{or}1}.\mathsf{C}(0)$
20 Return (crs, td)

Sim proof generator $\mathsf{S}^{0\text{or}1}.\mathsf{P}(\text{crs}, \text{td}, c)$:

21 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$; $(q, \gamma) \leftarrow \text{td}$
22 $p \leftarrow n/q$
23 If $(\mathbb{1}_{\mathbb{G}} \notin \{c, c^q, c^p\})$ then
24 $\quad A \leftarrow c$; $B \leftarrow cg^{-1}$
25 Else $A \leftarrow cg^{-1}$; $B \leftarrow c$
26 $a \leftarrow\!\$\ \mathbb{Z}_n^*$; $b \leftarrow a^{-1}$
27 $\text{pf}_1 \leftarrow A^a$; $\text{pf}_2 \leftarrow B^b$; $\text{pf}_3 \leftarrow \text{pf}_1^\gamma$
28 $\text{pf} \leftarrow (\text{pf}_1, \text{pf}_2, \text{pf}_3)$
29 Return pf

Claim validator $\mathsf{CV}^{0\text{or}1\text{-msg}}(\text{crs}, c, m)$:

30 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$
31 If $(m \notin \{0, 1\})$ then return false
32 Return $(\exists r \in \mathbb{Z} : c = g^m h^r)$

Ext CRS generator $\mathsf{E}^{0\text{or}1}.\mathsf{C}$:

33 $(\text{crs}, \text{td}) \leftarrow\!\$\ \mathsf{D\Pi}^{0\text{or}1}.\mathsf{C}(1)$
34 Return (crs, td)

Ext witness generator $\mathsf{E}^{0\text{or}1}.\mathsf{X}(\text{crs}, \text{td}, c, \text{pf})$:

35 $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$; $q \leftarrow \text{td}$
36 If $(c^q = \mathbb{1}_{\mathbb{G}})$ then $m \leftarrow 0$ else $m \leftarrow 1$
37 Return m

Figure 17: Left: Zero-or-One claim validator $\mathsf{CV}^{0\text{or}1}$ and dual-mode proof system $\mathsf{D\Pi}^{0\text{or}1}$ (three algorithms). The inputs to the proof generator must satisfy $c \in \mathbb{G}$ and $w \in \{0, 1\} \times \mathbb{Z}$. Right: Simulator S (two algorithms), a second claim validator $\mathsf{CV}^{0\text{or}1\text{-msg}}$, and extractor S (two algorithms), for Theorem 7.4.

---

computational soundness. The paper has, in this regard, no formal claim or proof. They seem to think it a consequence of the perfect soundness in mode 1 and the mode indistinguishability (our Theorem 7.2). However, it is not clear soundness transfers under mode indistinguishability, at least under our SND1 definition, as discussed in Section 4.6. As further discussed there, it does seem to transfer under a definition of soundness in the exclusion, rather than penalty, style. Which of these definitions of soundness is used in GOS [GOS06] is not clear to me. The definition they write is somewhat ambiguous in this regard.

Proof of Theorem 7.4: For **1**, let $\mu \in \{0, 1\}$. Suppose $(\text{crs}, \text{td}) \in [\mathsf{D\Pi}^{0\text{or}1}.\mathsf{C}(\mu)]$ and parse the CRS into its constituents, $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \text{crs}$. Suppose c, w satisfy $\mathsf{CV}^{0\text{or}1}(\text{crs}, c, w) = \mathsf{true}$, meaning $c = g^m h^r$ where $w \in \{0, 1\} \times \mathbb{Z}$ parses as $(m, r) \leftarrow w$. Let $s, t, u, \text{pf}_1, \text{pf}_2, \text{pf}_3$ be as at lines 10,11 of Figure 17. Then the condition at line 14 is not true, so this line does not return false. Now, making extensive use of Proposition 5.3, we have

$$\mathbf{e}(\text{pf}_1, \text{pf}_2) = \mathbf{e}(h^s, u^{rt}) = \mathbf{e}(h^r, u^{st}) = \mathbf{e}(h^r, u) = \mathbf{e}(h^r, g^{2m-1} h^r) \ .$$

Now consider the two choices for m separately. If $m = 0$ then $\mathbf{e}(h^r, g^{2m-1} h^r) = \mathbf{e}(h^r, g^{-1} h^r) =$

$\textsc{Init}()$: // Games G_0, G_1, G_2
1 $b \leftarrow\$ \{0,1\}$
2 $h \leftarrow\$ \mathrm{Gen}(\mathbb{G})$; $\gamma \leftarrow\$ \mathbb{Z}_n^*$; $g \leftarrow h^\gamma$
3 $\mathrm{crs} \leftarrow ((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h)$; Return crs

$\textsc{Fin}(b')$: // Games G_0, G_1, G_2
4 Return $(b' = 1)$

$\textsc{Pf}(c, w)$: // Game G_0
5 $(m, r) \leftarrow w$
6 $s \leftarrow\$ \mathbb{Z}_n^*$
7 $t \leftarrow s^{-1} \bmod n$; $u \leftarrow g^{2m-1}h^r$
8 $\mathrm{pf}_1 \leftarrow h^s$; $\mathrm{pf}_2 \leftarrow u^{rt}$; $\mathrm{pf}_3 \leftarrow g^s$
9 $\mathrm{pf} \leftarrow (\mathrm{pf}_1, \mathrm{pf}_2, \mathrm{pf}_3)$; Return pf

$\textsc{Pf}(c, w)$: // Game G_1
10 $(m, r) \leftarrow w$; $v \leftarrow \gamma m + r$
11 If $(v \notin \mathbb{Z}_n^*)$ then $v \leftarrow \gamma(m-1) + r$ // $v \in \mathbb{Z}_n^*$
12 $a \leftarrow\$ \mathbb{Z}_n^*$; $s \leftarrow av$
13 $t \leftarrow s^{-1} \bmod n$; $u \leftarrow g^{2m-1}h^r$
14 $\mathrm{pf}_1 \leftarrow h^s$; $\mathrm{pf}_2 \leftarrow u^{rt}$; $\mathrm{pf}_3 \leftarrow g^s$
15 $\mathrm{pf} \leftarrow (\mathrm{pf}_1, \mathrm{pf}_2, \mathrm{pf}_3)$; Return pf

$\textsc{Pf}(c, w)$: // Game G_2
16 $(m, r) \leftarrow w$; $v \leftarrow \gamma m + r$
17 If $(v \notin \mathbb{Z}_n^*)$ then $v \leftarrow \gamma(m-1) + r$ // $v \in \mathbb{Z}_n^*$
18 $A \leftarrow c$; $B \leftarrow cg^{-1}$
19 If $(c \notin \mathrm{Gen}(\mathbb{G}))$ then $A \leftarrow cg^{-1}$; $B \leftarrow c$ // $A \in \mathrm{Gen}(\mathbb{G})$
20 $a \leftarrow\$ \mathbb{Z}_n^*$; $b \leftarrow a^{-1}$; $s \leftarrow av$
21 $t \leftarrow s^{-1} \bmod n$; $u \leftarrow g^{2m-1}h^r$
22 $\mathrm{pf}_1 \leftarrow A^a$; $\mathrm{pf}_2 \leftarrow B^b$; $\mathrm{pf}_3 \leftarrow A^{a\gamma}$
23 $\mathrm{pf} \leftarrow (\mathrm{pf}_1, \mathrm{pf}_2, \mathrm{pf}_3)$; Return pf

Figure 18: Games for proof of part **3** of Theorem 7.4.

$\mathbf{e}(c, cg^{-1})$. Also if $m = 1$ then $\mathbf{e}(h^r, g^{2m-1}h^r) = \mathbf{e}(h^r, gh^r) = \mathbf{e}(gh^r, h^r) = \mathbf{e}(c, cg^{-1})$. So in either case we have $\mathbf{e}(\mathrm{pf}_1, \mathrm{pf}_2) = \mathbf{e}(c, cg^{-1})$, meaning $\mathrm{vf}_1 = \mathsf{true}$ at line 16. Also

$$\mathbf{e}(h, \mathrm{pf}_3) = \mathbf{e}(h, g^s) = \mathbf{e}(h^s, g) = \mathbf{e}(\mathrm{pf}_1, g) ,$$

so $\mathrm{vf}_2 = \mathsf{true}$ at line 17. So line 18 returns true.

For **2**, let $\mathrm{crs} \in [\Pi_1.\mathsf{C}]$ be the response to A's $\textsc{Init}$ query in game $\mathbf{G}^{\mathrm{snd1}}_{\Pi_1,\mathsf{TC}}$ of Figure 5, and parse it as $((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$. Let c, pf be a $\textsc{Vf}$ query of A . (The role of x in the game is played here by c.) Assume $c \in \mathbb{G}$ and $\mathrm{pf} \in \mathbb{G}^3$, for otherwise line 14 of Figure 17 returns false and we are done, and parse pf as at line 15. Define $f_g \colon \mathbb{G} \to \mathbb{T}$ by $f_g(A) = \mathbf{e}(A, g)$. Assume vf_1 at line 16 is true, for otherwise game $\mathbf{G}^{\mathrm{snd1}}_{\Pi_1,\mathsf{TC}}$ does not set win to true. Then

$$f_g(\mathrm{pf}_1^q) = f_g(\mathrm{pf}_1)^q = \mathbf{e}(h, \mathrm{pf}_3)^q = \mathbf{e}(h^q, \mathrm{pf}_3) = \mathbb{1}_{\mathbb{T}} .$$

Proposition 5.3 says that f_g is a group isomorphism, so it must be that $\mathrm{pf}_1^q = \mathbb{1}_{\mathbb{G}}$. Now assume vf_2 at line 17 is true, for otherwise game $\mathbf{G}^{\mathrm{snd1}}_{\Pi_1,\mathsf{CV}^{0\mathrm{or}1}}$ does not set win to true. Then

$$\mathbf{e}(c, cg^{-1})^q = \mathbf{e}(\mathrm{pf}_1, \mathrm{pf}_2)^q = \mathbf{e}(\mathrm{pf}_1^q, \mathrm{pf}_2) = \mathbf{e}(\mathbb{1}_{\mathbb{G}}, \mathrm{pf}_2) = \mathbb{1}_{\mathbb{T}} .$$

So condition **3** of Lemma 7.1 is met. The Lemma now says that condition **1** is met as well, which means $c \in \mathsf{TrCl}_{\mathsf{CV}^{0\mathrm{or}1}}(\mathrm{crs})$. So game $\mathbf{G}^{\mathrm{snd1}}_{\Pi_1,\mathsf{TC}}$ will not set win to true.

For **3**, consider the games in Figure 18. Assume adversary A makes no trivial queries, so that any $\textsc{Pf}(c, w)$ query satisfies $\mathsf{CV}^{0\mathrm{or}1}(\mathrm{crs}, c, w) = \mathsf{true}$ —that is, $c = g^m h^r \in \mathbb{G}$ where $w = (m, r) \in \{0,1\} \times \mathbb{Z}$— allowing us to eliminate line 5 of game $\mathbf{G}^{\mathrm{zk}}_{\Pi_0,\mathsf{CV}^{0\mathrm{or}1},\mathsf{S}}$. Algorithms $\Pi_0.\mathsf{C}$ and $\mathsf{S}^{0\mathrm{or}1}.\mathsf{C}$ create crs the same way, via $\mathsf{D}\Pi^{0\mathrm{or}1}.\mathsf{C}(0)$, so the games do as well, retaining the trapdoor information q, γ. In game G_0, oracle $\textsc{Pf}$ computes its replies using $\mathsf{D}\Pi^{0\mathrm{or}1}.\mathsf{P}$, while in game G_2 it does so using

$\mathsf{S}^{0\text{or}1}.\mathsf{P}$, so

$$\mathbf{Adv}^{\text{zk}}_{\Pi_0,\mathsf{CV}^{0\text{or}1},\mathsf{S}}(\mathrm{A}) = \Pr[\mathrm{G}_0(\mathrm{A})] - \Pr[\mathrm{G}_2(\mathrm{A})]$$
$$= (\Pr[\mathrm{G}_0(\mathrm{A})] - \Pr[\mathrm{G}_1(\mathrm{A})]) + (\Pr[\mathrm{G}_1(\mathrm{A})] - \Pr[\mathrm{G}_2(\mathrm{A})]) \; .$$

We claim that in G_1, either $\gamma m + r \in \mathbb{Z}_n^*$ or $\gamma(m-1)+r \in \mathbb{Z}_n^*$, meaning either $\gcd(\gamma m + r, n) = 1$ or $\gcd(\gamma(m-1)+r, n) = 1$. (Justifying this is left as an exercise.) This means v is always in $\mathbb{Z}_n^*$ after line 11. So line 12 is equivalent to line 6 with regard to the distribution of s. (Why is it important for this that $v \in \mathbb{Z}_n^*$?) Thus $\Pr[\mathrm{G}_0(\mathrm{A})] = \Pr[\mathrm{G}_1(\mathrm{A})]$. To complete the proof we show that $\Pr[\mathrm{G}_1(\mathrm{A})] = \Pr[\mathrm{G}_2(\mathrm{A})]$. This is done by showing that $(A^a, B^b, A^{a\gamma}) = (h^s, u^{rt}, g^s)$ in game G_2. We consider two cases. The first is that $c \in \mathrm{Gen}(\mathbb{G})$, which happens iff $\gamma m + r \in \mathbb{Z}_n^*$. Since $g = h^\gamma$, we have $c = g^m h^r = h^{\gamma m + r} = h^v$. So

$$A^a = c^a = h^{av} = h^s \; .$$

We show that $B^b = u^{rt}$ by showing that the discrete logarithms of these quantities in base h are the same. In the following, arithmetic is modulo n, and we use the fact that $m(m-1) = 0$, which is true because $m \in \{0, 1\}$:

$$\begin{aligned}\mathrm{dlog}_{\mathbb{G},h}(B^b) &= b \cdot \mathrm{dlog}_{\mathbb{G},h}(cg^{-1}) = s^{-1}v \cdot \mathrm{dlog}_{\mathbb{G},h}(h^{\gamma(m-1)+r}) = tv(\gamma(m-1)+r)\\ &= (\gamma(m-1)+r)(\gamma m + r)t = (\gamma m + \gamma(m-1) + r)rt = (\gamma(2m-1)+r)rt\\ &= rt \cdot \mathrm{dlog}_{\mathbb{G},h}(h^{\gamma(2m-1)+r}) = rt \cdot \mathrm{dlog}_{\mathbb{G},h}(g^{2m-1}h^r) = \mathrm{dlog}_{\mathbb{G},h}(u^{rt}) \; .\end{aligned}$$

Also

$$A^{a\gamma} = c^{a\gamma} = h^{av\gamma} = g^{av} = g^s \; .$$

The second case is that $c \notin \mathrm{Gen}(\mathbb{G})$, which happens iff $\gamma m + r \notin \mathbb{Z}_n^*$. We have $cg^{-1} = g^{m-1}h^r = h^{\gamma(m-1)+r} = h^v$. So

$$A^a = (cg^{-1})^a = h^{av} = h^s \; .$$

The other checks are left as an exercise. ▌

Theorem 7.4 covers the claims made in GOS [GOS06]. We now turn to some further claims that they don't make but we think can be established.

Theorem 7.5, below, says Π_1 has perfect extractability for $\mathsf{CV}^{0\text{or}1\text{-msg}}$, the alternative claim validator shown on the right in Figure 17, showing it is a proof of knowledge of the message underlying a ciphertext, even if not of the underlying randomness. This shows the value of establishing the proof of knowledge for a claim validator different from the main one, something that does not seem to explicitly be in the literature, at least to best of my (limited) knowledge. In particular [GOS06] has no claim about the "Zero-or-One" system being a proof of knowledge, possibly because it isn't for $\mathsf{CV}^{0\text{or}1}$. (They do show the proof of knowledge property for their circuit-satisfiability system, which we will get to in a bit.)

Theorem 7.5 *Let* GG *be a group generator. Let* $\mathsf{D\Pi}^{0\text{or}1}$ *be the associated zero-or-one dual-mode proof system, and* $\mathsf{CV}^{0\text{or}1}, \mathsf{CV}^{0\text{or}1\text{-msg}}$ *the associated claim validators, as per the left of Figure 17. Let* $\mathsf{TC} = \mathsf{TrCl}_{\mathsf{CV}^{0\text{or}1}} = \mathsf{TrCl}_{\mathsf{CV}^{0\text{or}1\text{-msg}}}$ *be the true-claim language common to both claim validators. Let* Π_1, Π_0 *be the proof systems induced by* $\mathsf{D\Pi}^{0\text{or}1}$*. Then:*

1. Π_1 *satisfies perfect extractability for* $\mathsf{CV}^{0\text{or}1\text{-msg}}$ *relative to the extractor* $\mathsf{E}^{0\text{or}1}$ *shown on the*

Claim validator $\mathsf{CV}^{\mathrm{cs}}(\mathrm{crs}, \mathsf{C}, w)$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$; $(m, r) \leftarrow w$
$\boldsymbol{w} \leftarrow \mathsf{Ev}(\mathsf{C}, w)$
Return ($\boldsymbol{w}[\mathsf{C.In} + \mathsf{C.Gt}] = 1$)

CRS generator $\mathsf{D\Pi}^{\mathrm{cs}}.\mathsf{C}(\mu)$: // $\mu \in \{0,1\}$

$(n, q, \mathbb{G}, \mathbb{T}, \mathbf{e}) \leftarrow\!\!\$\ \mathsf{GG}$
If $(\mu = 1)$ then $g \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$; $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G}_q)$; $\mathrm{td} \leftarrow q$
Else $h \leftarrow\!\!\$\ \mathrm{Gen}(\mathbb{G})$; $\gamma \leftarrow\!\!\$\ \mathbb{Z}_n^*$; $g \leftarrow h^{\gamma}$; $\mathrm{td} \leftarrow (q, \gamma)$
$\mathrm{crs} \leftarrow ((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h)$; Return $(\mathrm{crs}, \mathrm{td})$

Proof generator $\mathsf{D\Pi}^{\mathrm{cs}}.\mathsf{P}(\mathrm{crs}, \mathsf{C}, w)$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$
$\boldsymbol{m} \leftarrow \mathsf{Ev}(\mathsf{C}, w)$; $\ell \leftarrow \mathsf{C.In} + \mathsf{C.Gt}$
$\boldsymbol{c}[\ell] \leftarrow g$; $\boldsymbol{r}[\ell] \leftarrow 0$
For $i = 1, \ldots, \ell - 1$ do
  $\boldsymbol{r}[i] \leftarrow\!\!\$\ \mathbb{Z}_n^*$; $\boldsymbol{c}[i] \leftarrow g^{\boldsymbol{m}[i]} h^{\boldsymbol{r}[i]}$
  $\boldsymbol{p}[i] \leftarrow\!\!\$\ \mathsf{D\Pi}^{0\mathrm{or}1}.\mathsf{P}(\mathrm{crs}, \boldsymbol{c}[i], (\boldsymbol{m}[i], \boldsymbol{r}[i]))$
For $\mathtt{G} = \mathsf{C.In} + 1, \ldots, \ell$ do
  $m \leftarrow \mathrm{NAND}(\ \boldsymbol{w}[\mathsf{C.I1}(\mathtt{G})], \boldsymbol{w}[\mathsf{C.I2}(\mathtt{G})]\ )$
  $c \leftarrow \boldsymbol{c}[\mathsf{C.I1}(\mathtt{G})] \cdot \boldsymbol{c}[\mathsf{C.I2}(\mathtt{G})] \cdot \boldsymbol{c}[\mathtt{G}]^2 \cdot g^{-2}$
  $r \leftarrow \boldsymbol{r}[\mathsf{C.I1}(\mathtt{G})] + \boldsymbol{r}[\mathsf{C.I2}(\mathtt{G})] + 2\boldsymbol{r}[\mathtt{G}]$
  $\boldsymbol{p}[\ell + \mathtt{G}] \leftarrow\!\!\$\ \mathsf{D\Pi}^{0\mathrm{or}1}.\mathsf{P}(\mathrm{crs}, c, (m, r))$
Return $(\boldsymbol{c}, \boldsymbol{p})$

Proof verifier $\mathsf{D\Pi}^{\mathrm{cs}}.\mathsf{V}(\mathrm{crs}, \mathsf{C}, (\boldsymbol{c}, \boldsymbol{p}))$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$; $\ell \leftarrow \mathsf{C.In} + \mathsf{C.Gt}$
$\boldsymbol{v}[\ell] \leftarrow (\boldsymbol{c}[\ell] = g)$
For $i = 1, \ldots, \ell - 1$ do
  $\boldsymbol{v}[i] \leftarrow \mathsf{D\Pi}^{0\mathrm{or}1}.\mathsf{V}(\mathrm{crs}, \boldsymbol{c}[i], \boldsymbol{p}[i])$
For $\mathtt{G} = \mathsf{C.In} + 1, \ldots, \ell$ do
  $c \leftarrow \boldsymbol{c}[\mathsf{C.I1}(\mathtt{G})] \cdot \boldsymbol{c}[\mathsf{C.I2}(\mathtt{G})] \cdot \boldsymbol{c}[\mathtt{G}]^2 \cdot g^{-2}$
  $\boldsymbol{v}[\ell + \mathtt{G}] \leftarrow \mathsf{D\Pi}^{0\mathrm{or}1}.\mathsf{V}(\mathrm{crs}, c, \boldsymbol{p}[\mathtt{G}])$
Return ($\forall\, i\, :\, \boldsymbol{v}[i] = \mathsf{true}$)

Sim CRS generator $\mathsf{S}^{\mathrm{cs}}.\mathsf{C}$:

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{D\Pi}^{\mathrm{cs}}.\mathsf{C}(0)$
Return $(\mathrm{crs}, \mathrm{td})$

Sim proof generator $\mathsf{S}^{\mathrm{cs}}.\mathsf{P}(\mathrm{crs}, \mathrm{td}, \mathsf{C})$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$; $(q, \gamma) \leftarrow \mathrm{td}$
$\ell \leftarrow \mathsf{C.In} + \mathsf{C.Gt}$
$\boldsymbol{w}[\ell] \leftarrow 1$; $\boldsymbol{c}[\ell] \leftarrow g$
For $i = 1, \ldots, \ell - 1$ do
  $\boldsymbol{w}[i] \leftarrow 0$; $\boldsymbol{r}[i] \leftarrow\!\!\$\ \mathbb{Z}_n^*$; $\boldsymbol{c}[i] \leftarrow g^{\boldsymbol{w}[i]} h^{\boldsymbol{r}[i]}$
  $\boldsymbol{p}[i] \leftarrow\!\!\$\ \mathsf{S}^{0\mathrm{or}1}.\mathsf{P}(\mathrm{crs}, \mathrm{td}, \boldsymbol{c}[i])$
For $\mathtt{G} = \mathsf{C.In} + 1, \ldots, \ell$ do
  $c \leftarrow \boldsymbol{c}[\mathsf{C.I1}(\mathtt{G})] \cdot \boldsymbol{c}[\mathsf{C.I2}(\mathtt{G})] \cdot \boldsymbol{c}[\mathtt{G}]^2 \cdot g^{-2}$
  $\boldsymbol{p}[\ell + \mathtt{G}] \leftarrow\!\!\$\ \mathsf{S}^{0\mathrm{or}1}.\mathsf{P}(\mathrm{crs}, \mathrm{td}, c)$
Return $(\boldsymbol{c}, \boldsymbol{p})$

Ext CRS generator $\mathsf{E}^{\mathrm{cs}}.\mathsf{C}$:

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{D\Pi}^{\mathrm{cs}}.\mathsf{C}(1)$
Return $(\mathrm{crs}, \mathrm{td})$

Ext witness generator $\mathsf{E}^{\mathrm{cs}}.\mathsf{X}(\mathrm{crs}, \mathrm{td}, \mathsf{C}, (\boldsymbol{c}, \boldsymbol{p}))$:

$((n, \mathbb{G}, \mathbb{T}, \mathbf{e}), g, h) \leftarrow \mathrm{crs}$
For $i = 1, \ldots, \mathsf{C.In}$ do
  $w[i] \leftarrow \mathsf{E}^{0\mathrm{or}1}.\mathsf{X}(\mathrm{crs}, \mathrm{td}, \boldsymbol{c}[i], \boldsymbol{p}[i])$
Return w

Figure 19: Left: Circuit satisfiability claim validator $\mathsf{CV}^{\mathrm{cs}}$ and dual-mode proof system $\mathsf{D\Pi}^{\mathrm{cs}}$ (three algorithms). The inputs to the proof generator are a circuit C and an assignment $w \in \{0,1\}^{\mathsf{C.In}}$ to its variables. Right: Simulator S (two algorithms) and extractor S (two algorithms), for Theorem 7.7.

---

right of Figure 17: $\mathbf{Adv}^{\mathrm{xt1}}_{\Pi_1, \mathsf{CV}^{0\mathrm{or}1\text{-}\mathrm{msg}}, \mathsf{E}^{0\mathrm{or}1}}(A) = 0$ *for all adversaries* A.

Now we turn to the circuit-satisfiability proof system from [GOS06]. It reduces to the zero-or-one system via the following Lemma.

Lemma 7.6 [GOS06] *Let* $b_0, b_1, b_2 \in \{0,1\}$. *Then*

$$b_2 = \mathrm{NAND}(b_0, b_1) \quad \textit{iff} \quad b_0 + b_1 + 2b_2 - 2 \in \{0,1\}\ .$$

Proof of Lemma 7.6: The table below enumerates all possible values of b_0, b_1, b_2 and for each, computes both $\mathrm{NAND}(b_0, b_1)$ and $b_0 + b_1 + 2b_2 - 2$.

b_0	b_1	b_2	$\mathrm{NAND}(b_0, b_1)$	$b_0 + b_1 + 2b_2 - 2$
0	0	0	1	-2
0	1	0	1	-1
1	0	0	1	-1
1	1	0	0	0
0	0	1	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	2

The claim of the lemma can be verified from the table. ▮

Consider a circuit that is a single NAND gate. Its input wires are $1, 2$ and its output wire is 3. The prover has an assignment $w \in \{0,1\}^2$ to the inputs and wants to create a correct encrypted version of the gate. The prover will annotate each wire with a ciphertext:

$$\boldsymbol{c}[1] = g^{w[1]} h^{r_1}$$
$$\boldsymbol{c}[2] = g^{w[2]} h^{r_2}$$
$$\boldsymbol{c}[3] = g^{w[3]} h^{r_3} \; ,$$

where $w[3] = \mathrm{NAND}(w[1], w[2])$ and the prover picks $r_1, r_2, r_3 \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_n^*$. Using the zero-or-one system, the prover proves in zero-knowledge that the messages underlying the three ciphertexts are bits. (Thus, the value of the bits is not revealed.) Let $\boldsymbol{p}[1], \boldsymbol{p}[2], \boldsymbol{p}[3]$ denote these three proofs, each constructed by running the prover of Figure 17. Now the prover will also prove that the bit underlying $\boldsymbol{c}[3]$ is indeed the NAND of the bits underlying the other two ciphertexts. According to Lemma 7.6, it needs to show that $x = w[1] + w[2] + 2w[3] - 2 \in \{0, 1\}$. It forms the ciphertext

$$c = \boldsymbol{c}[1] \cdot \boldsymbol{c}[2] \cdot \boldsymbol{c}[3]^2 \cdot g^{-2} = g^{w[1]+w[2]+2w[3]-2} h^{r_1+r_2+2r_3} = g^x h^r$$

where $r \leftarrow r_1 + r_2 + 2r_3$. Now it uses the zero-or-one system to generate a proof $\boldsymbol{p}[4]$ that c encrypts either a 0 or a 1. The proof that it outputs is $(\boldsymbol{c}, \boldsymbol{p})$.

Again we cast it as a dual-mode proof system. The full system, and the associated claim validator, are on the left in Figure 19.

Theorem 7.7 *Let* GG *be a group generator. Let* $\mathsf{D\Pi}^{\mathrm{cs}}$ *be the circuit satisfiability dual-mode proof system, and* $\mathsf{CV}^{\mathrm{cs}}$ *the associated claim validator, as per the left of Figure 19. Let* $\mathsf{TC} = \mathsf{TrCl}_{\mathsf{CV}^{\mathrm{cs}}}$ *be the true-claim language associated to* $\mathsf{CV}^{\mathrm{cs}}$*. Let* $\mathsf{\Pi}_1, \mathsf{\Pi}_0$ *be the proof systems induced by* $\mathsf{D\Pi}^{\mathrm{cs}}$*. Then:*

1. $\mathsf{\Pi}_1$ *and* $\mathsf{\Pi}_0$ *satisfy perfect completeness for* $\mathsf{CV}^{\mathrm{cs}}$
2. $\mathsf{\Pi}_1$ *satisfies perfect soundness for* TC*:* $\mathbf{Adv}^{\mathrm{snd1}}_{\mathsf{\Pi}_1,\mathsf{TC}}(\mathrm{A}) = 0$ *for all adversaries* A.
3. $\mathsf{\Pi}_0$ *satisfies perfect zero knowledge for* $\mathsf{CV}^{\mathrm{cs}}$ *relative to the simulator* S^{cs} *shown on the right of Figure 17:* $\mathbf{Adv}^{\mathrm{zk}}_{\mathsf{\Pi}_0,\mathsf{CV}^{\mathrm{cs}},\mathsf{S}^{\mathrm{cs}}}(\mathrm{A}) = 0$ *for all adversaries* A.
4. $\mathsf{\Pi}_1$ *satisfies perfect extractability for* $\mathsf{CV}^{\mathrm{cs}}$ *relative to the extractor* E^{cs} *shown on the right of Figure 17:* $\mathbf{Adv}^{\mathrm{xt1}}_{\mathsf{\Pi}_1,\mathsf{CV}^{\mathrm{cs}},\mathsf{E}^{\mathrm{cs}}}(\mathrm{A}) = 0$ *for all adversaries* A.

Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}$	Game $\mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}}$
$\textsc{Init}()$:	$\textsc{Init}()$:
$(sk, vk) \leftarrow\!\!\$\ \mathsf{DS.K}$; Return vk	$(sk, vk) \leftarrow\!\!\$\ \mathsf{DS.K}$; Return vk
$\textsc{Sign}(m)$:	$\textsc{Sign}(m)$:
$\sigma \leftarrow\!\!\$\ \mathsf{DS.S}(sk, vk, m)$; $S \leftarrow S \cup \{m\}$	$\sigma \leftarrow\!\!\$\ \mathsf{DS.S}(sk, vk, m)$; $S \leftarrow S \cup \{(m, \sigma)\}$
Return σ	Return σ
$\textsc{Fin}(m, \sigma)$:	$\textsc{Fin}(m, \sigma)$:
$\mathrm{vf} \leftarrow \mathsf{DS.V}(vk, m, \sigma)$	$\mathrm{vf} \leftarrow \mathsf{DS.V}(vk, m, \sigma)$
Return (vf and $(m \notin S)$)	Return (vf and $((m, \sigma) \notin S)$)

Figure 20: Games defining UF and SUF security of a signature scheme DS.

8 Signatures from NIZKs

We explore different ways to build a signature scheme from a NIZK proof system. Depending on the strength assumed from the NIZK, we have different constructions that make different assumptions beyond the NIZK.

8.1 Signature definitions

A (digital) signature scheme DS specifies algorithms for key-generation, signing and verifying, as follows. Via $(sk, vk) \leftarrow\!\!\$\ \mathsf{DS.K}$, the signer generates a secret signing key sk and public verification key vk. Via $\sigma \leftarrow\!\!\$\ \mathsf{DS.S}(sk, vk, m)$, the signer generates a signature of a message $m \in \{0,1\}^*$. Via $\mathrm{vf} \leftarrow \mathsf{DS.V}(vk, m, \sigma)$, the verifier deterministically generates a boolean decision as to the validity of σ. Correctness requires that $\mathsf{DS.V}(vk, m, \sigma) = \mathtt{true}$ for all $\sigma \in [\mathsf{DS.S}(sk, vk, m)]$, all $(sk, vk) \in [\mathsf{DS.K}]$ and all $m \in \{0,1\}^*$.

The security metrics for a signature scheme are unforgeability (UF) [GMR88] and strong unforgeability (SUF). The games are in Figure 20. We let $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathrm{A})]$ and $\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}}(\mathrm{A})]$ be the corresponding advantages of adversary A. The following says that SUF security implies UF security.

Proposition 8.1 *Let* DS *be a signature scheme and* A *an adversary. Then*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathrm{A}) \le \mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}}(\mathrm{A}) \; .$$

Exercise 8.2 *Prove Proposition 8.1.*

Exercise 8.3 *Show by counter-example that UF does not imply SUF.*

8.2 Construction from ZK+XT2

Let $\mathsf{F}\colon \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \{0,1\}^*$ be a family of functions that we assume is one-way as defined in Section 3, and let CV be the claim validator of Figure 21. Let Π be a proof system that satisfies (completeness and) zkxt for CV. Let DS_1 be the signature scheme whose algorithms are shown in

Claim validator $\mathsf{CV}(\mathrm{crs}, (K, Y, m), X')$:

Return ($\mathsf{F}(K, X') = Y$)

Key-generation algorithm $\mathsf{DS}_1.\mathsf{K}$:

$K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$

$sk \leftarrow\!\!\$\ \mathsf{F.D}$; $Y \leftarrow \mathsf{F}(K, sk)$

$\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$; $vk \leftarrow (\mathrm{crs}, Y)$

Return (sk, vk)

Signing algorithm $\mathsf{DS}_1.\mathsf{S}(sk, vk, m)$:

$(K, Y) \leftarrow vk$

$\sigma \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, (K, Y, m), sk)$

Return σ

Verifying algorithm $\mathsf{DS}_1.\mathsf{V}(vk, m, \sigma)$:

$(K, Y) \leftarrow vk$

Return $\Pi.\mathsf{V}(\mathrm{crs}, (K, Y, m), \sigma)$

Adversary A_{ow}:

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$; $(K, Y) \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{ow}}_{\mathsf{F}}.\textsc{Init}$; $vk \leftarrow (\mathrm{crs}, Y)$

$\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:

Return vk

$\textsc{Sign}(m)$:

$\sigma \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (K, Y, m))$; Return σ

$\textsc{Fin}(m, \sigma)$:

$X' \leftarrow \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, (K, Y, m), \sigma)$

$\mathbf{G}^{\mathrm{ow}}_{\mathsf{F}}.\textsc{Fin}(X')$

Adversary $\mathrm{A}_{\mathrm{zkxt}}$:

$\mathrm{crs} \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Init}$

$K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $sk \leftarrow\!\!\$\ \mathsf{F.D}$; $Y \leftarrow \mathsf{F}(K, sk)$; $vk \leftarrow (\mathrm{crs}, Y)$

$\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:

Return vk

$\textsc{Sign}(m)$:

$\sigma \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Pf}((K, Y, m), sk)$

$S \leftarrow S \cup \{(m, \sigma)\}$; Return σ

$\textsc{Fin}(m, \sigma)$:

$\mathrm{vf} \leftarrow \Pi.\mathsf{V}(\mathrm{crs}, (K, Y, m), \sigma)$

If ($(\mathrm{vf} = \mathsf{false})$ or $((m, \sigma) \in S)$) then $\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(0)$

$\mathrm{ex} \leftarrow \mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Ex}((K, Y, m), \sigma)$

If (ex) then $b' \leftarrow 1$ else $b' \leftarrow 0$

$\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(b')$

Figure 21: Left: Signature scheme DS_1. Right: Adversaries for Theorem 8.4.

Figure 21. Theorem 8.4 says that DS_1 is SUF secure. This shows that if we have a NIZK that meets the joint ZKXT notion (which is equivalent to ZK+XT2) then it is easy to build a secure signature scheme.

The theorem statement, being concrete, does not have language of the form above with regard to F, Π and what is assumed about them. The one-wayness assumption on F emerges in the term involving the ow-advantage of A_{ow} in Equation (6). The ZKXT is captured by assuming a simulator S, the zkxt-advantage of an adversary $\mathrm{A}_{\mathrm{zkxt}}$ relative to it, showing up again in Equation (6), being viewed as small.

This construction is from [DHLW10, BMT14]. The proofs there used ZK and XT2 separately. Our proof from ZKXT simplifies the prior ones.

In the Theorem statement, $\mathrm{T}(A)$ refers to the running time of an algorithm A.

Theorem 8.4 *Let* $\mathsf{F}\colon \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \{0,1\}^*$ *be a family of functions and let* CV *be the claim validator of Figure 21. Let* Π *be a proof system and* S *a simulator for it. Let* DS_1 *be the signature*

Game G_0

$\text{INIT}()$:
1 $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $sk \leftarrow\!\!\$\ \mathsf{F.D}$; $Y \leftarrow \mathsf{F}(K, sk)$
2 $\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$
3 $vk \leftarrow (\mathrm{crs}, Y)$; Return vk

$\text{SIGN}(m)$:
4 $\sigma \leftarrow\!\!\$\ \mathsf{\Pi.P}(\mathrm{crs}, (K, Y, m), sk)$
5 $S \leftarrow S \cup \{(m, \sigma)\}$; Return σ

$\text{FIN}(m, \sigma)$:
6 $\mathrm{vf} \leftarrow \mathsf{\Pi.V}(\mathrm{crs}, (K, Y, m), \sigma)$
7 Return (vf and $((m, \sigma) \notin S)$)

Game G_1

$\text{INIT}()$:
1 $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $sk \leftarrow\!\!\$\ \mathsf{F.D}$; $Y \leftarrow \mathsf{F}(K, sk)$
2 $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$
3 $vk \leftarrow (\mathrm{crs}, Y)$; Return vk

$\text{SIGN}(m)$:
4 $\sigma \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (K, Y, m))$
5 $S \leftarrow S \cup \{(m, \sigma)\}$; Return σ

$\text{FIN}(m, \sigma)$:
6 $\mathrm{vf} \leftarrow \mathsf{\Pi.V}(\mathrm{crs}, (K, Y, m), \sigma)$
7 If ($(\mathrm{vf} = \mathsf{false})$ or $((m, \sigma) \in S)$) then return false
8 $X' \leftarrow \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, (K, Y, m), \sigma)$
9 Return ($\mathsf{F}(K, X') = Y$)

Figure 22: Games for proof of Theorem 8.4.

scheme whose algorithms are shown in Figure 21. Let A_{ds} *be an adversary making* q *queries to its* SIGN *oracle. Then we can construct adversaries* A_{ow} *and* $\mathrm{A}_{\mathrm{zkxt}}$ *(shown explicitly in Figure 21) such that*

$$\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}_1}(\mathrm{A}_{\mathrm{ds}}) \le \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(\mathrm{A}_{\mathrm{ow}}) + \mathbf{Adv}^{\mathrm{zkxt}}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zkxt}}) \ . \tag{6}$$

The running time of A_{ow} *is that of* A_{ds} *plus* $\mathcal{O}(q \cdot \mathrm{T}(\mathsf{S.P}) + \mathrm{T}(\mathsf{S.C}) + \mathrm{T}(\mathsf{S.X}))$*. The running time of* $\mathrm{A}_{\mathrm{zkxt}}$ *is that of* A_{ds} *plus* $\mathcal{O}(\mathrm{T}(\mathsf{F}))$*.*

How does the choice of simulator influence security? The running time of the constructed algorithms depends on the running time of the simulator algorithms. Thus, the more efficient the simulator, the more security we have shown for the signature scheme. This tells us that improving simulator runtime is valuable.

The notation $\mathrm{T}(A)$ used for various algorithms A, above, fails to reflect that this time may depend also on the lengths of the inputs provided to the algorithms. For this reason, amongst others, the running-time estimates for the constructed adversaries given in the theorem statement should be viewed as rough indicators only. It is because of this deficiency in the statement that the theorem points to explicit, pseudocode constructions of the adversaries, which become the final reference points for the precise running times. One may ask, why not give precise running time estimates in the theorems? Finding language for this is hard because the inputs to the algorithms are random variables, and their lengths vary across the executions involved. It would be nice to find language that allows such statements to be made in simple ways.

Proof of Theorem 8.4: Consider the games of Figure 22. At lines 2,4, game G_1 switches to using the simulator. In FIN, it runs the extractor. We have

$$\begin{aligned} \mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}_1}(\mathrm{A}_{\mathrm{ds}}) &= \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] \\ &= \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] + (\Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})]) \ . \end{aligned}$$

We claim to have designed the Figure 21 adversaries so that

$$\Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F}}(\mathrm{A}_{\mathrm{ow}}) \tag{7}$$

$$\Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zkxt}}) \;. \tag{8}$$

Verifying the above two equations would conclude the proof, so let's do that.

Adversary A_{ow} answers A_{ds}'s queries as does game G_1. It calls $\mathsf{S.X}$ whenever game G_1 does. (Possibly more often because it omits the checks done by the game, but this can only increase its advantage.) This justifies Equation (7).

Let b denote the challenge bit chosen at random in game $\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}$, and let b' denote the query of $\mathrm{A}_{\mathrm{zkxt}}$ to $\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}$. Then

$$\mathbf{Adv}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zkxt}}) = \Pr\left[\, b' = 1 \;|\; b = 1 \,\right] - \Pr\left[\, b' = 1 \;|\; b = 0 \,\right] \;,$$

where the probabilities are in the execution of game $\mathbf{G}^{\mathrm{zkxt}}_{\Pi,\mathsf{CV},\mathsf{S}}$ with adversary $\mathrm{A}_{\mathrm{zkxt}}$. Now we claim

$$\begin{aligned} \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] &= \Pr\left[\, b' = 1 \;|\; b = 1 \,\right] \\ \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] &= \Pr\left[\, b' = 1 \;|\; b = 0 \,\right] \;. \end{aligned}$$

If $b = 1$ then $\mathrm{ex} = \mathrm{vf} = \mathsf{true}$, which justifies the first. If $b = 0$ then ex equals $\mathsf{CV}(\mathrm{crs}, (K, Y, m), X')$, so is true iff $\mathsf{F}(K, X') = Y$, which justifies the second. Applying Lemma 3.2, this justifies Equation (8). ∎

8.3 Construction from ZK+SND1

Let $\mathsf{F}\colon \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \{0,1\}^*$ be a family of functions that we assume is uf-secure as defined in Section 3. Let CS be a commitment scheme with $\mathsf{CS.M} = \{0,1\}^{\mathsf{F.kl}}$, meaning keys for F are messages to which one can commit. We assume CS is *perfectly* binding and computationally hiding. Let CV be the claim validator of Figure 23. Let Π be a proof system that satisfies completeness, SND1 for $\mathsf{TrCl}_{\mathsf{CV}}$ and ZK for CV. Let DS_2 be the signature scheme whose algorithms are shown in Figure 23. Theorem 8.5 says that DS_2 is UF secure. This shows that if we have a NIZK that meets SND1+ZK then one can build a secure signature scheme, even if with more work compared to building it from the stronger ZKXT.

The construction and proof are from [BG90]. We have made some small changes: The assumption on F is reduced from PRF-security to UF-security and the hiding requirement on CS has been weakened.

Theorem 8.5 *Let $\mathsf{F}\colon \{0,1\}^{\mathsf{F.kl}} \times \mathsf{F.D} \to \{0,1\}^*$ be a family of functions. Let CS be a perfectly binding commitment scheme with $\mathsf{CS.M} = \{0,1\}^{\mathsf{F.kl}}$. Let CV be the claim validator of Figure 23. Let Π be a proof system and S a simulator for it. Let DS_2 be the signature scheme whose algorithms are shown in Figure 23. Let A_{ds} be an adversary making q queries to its* $\textsc{Sign}$ *oracle. Then we can construct adversaries $\mathrm{A}_{\mathrm{snd1}}, \mathrm{A}_{\mathrm{zk}}, \mathrm{A}_{\mathrm{hide}}, \mathrm{A}_{\mathrm{uf}}$ (shown explicitly in Figures 23 and 24) such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}_2}(\mathrm{A}_{\mathrm{ds}}) \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}}) + \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) + \mathbf{Adv}^{\mathrm{hide}}_{\mathsf{CS}}(\mathrm{A}_{\mathrm{hide}}) + \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{F}}(\mathrm{A}_{\mathrm{uf}}) \;. \tag{9}$$

The running time of A_{uf} is that of A_{ds} plus $\mathcal{O}(q \cdot \mathrm{T}(\mathsf{S.P}) + \mathrm{T}(\mathsf{S.C}) + \mathrm{T}(\mathsf{CS.C}) + \mathrm{T}(\mathsf{CS.P}))$. The running time of A_{zk} is that of A_{ds} plus $\mathcal{O}(\mathrm{T}(\mathsf{F}))$. The running time of ...

Claim validator $\mathsf{CV}(\mathrm{crs}, (cp, c, Y, m), (K, d))$:

$d_1 \leftarrow (\mathsf{CS.C}(cp, K, d) = c)$
$d_2 \leftarrow (Y = \mathsf{F}(K, m))$
Return $(d_1$ and $d_2)$

Key-generation algorithm $\mathsf{DS}_2.\mathsf{K}$:

$cp \leftarrow\$ \mathsf{CS.P}$; $K \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$
$d \leftarrow\$ \mathsf{CS.D}$; $c \leftarrow \mathsf{CS.C}(cp, K, d)$
$\mathrm{crs} \leftarrow\$ \Pi.\mathsf{C}$
$sk \leftarrow (K, d)$; $vk \leftarrow (\mathrm{crs}, cp, c)$
Return (sk, vk)

Signing algorithm $\mathsf{DS}_2.\mathsf{S}(sk, vk, m)$:

$(\mathrm{crs}, cp, c) \leftarrow vk$; $(K, d) \leftarrow sk$
$Y \leftarrow \mathsf{F}(K, m)$
$\mathrm{pf} \leftarrow\$ \Pi.\mathsf{P}(\mathrm{crs}, (cp, c, Y, m), (K, d))$
Return (Y, pf)

Verifying algorithm $\mathsf{DS}_2.\mathsf{V}(vk, m, \sigma)$:

$(\mathrm{crs}, cp, c) \leftarrow vk$; $(Y, \mathrm{pf}) \leftarrow \sigma$
Return $\Pi.\mathsf{V}(\mathrm{crs}, (cp, c, Y, m), \mathrm{pf})$

Adversary A_{uf}:

$cp \leftarrow\$ \mathsf{CS.P}$; $K' \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\$ \mathsf{CS.D}$
$c \leftarrow \mathsf{CS.C}(cp, K', d)$; $(\mathrm{crs}, \mathrm{td}) \leftarrow\$ \mathsf{S.C}$; $vk \leftarrow (\mathrm{crs}, cp, c)$
$\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}.\textsc{Init}$; $\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:
Return vk

$\textsc{Sign}(m)$:
$Y \leftarrow \mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}.\textsc{Fn}(m)$; $\mathrm{pf} \leftarrow\$ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (cp, c, Y, m))$
Return (Y, pf)

$\textsc{Fin}(m, \sigma)$:
$(Y, \mathrm{pf}) \leftarrow \sigma$; $\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}.\textsc{Fin}(m, Y)$

Adversary A_{zk}:

$cp \leftarrow\$ \mathsf{CS.P}$; $K \leftarrow\$ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\$ \mathsf{CS.D}$
$c \leftarrow \mathsf{CS.C}(cp, K, d)$; $\mathrm{crs} \leftarrow\$ \mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Init}$
$vk \leftarrow (\mathrm{crs}, cp, c)$; $\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:
Return vk

$\textsc{Sign}(m)$:
$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\$ \mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Pf}((cp, c, Y, m), (K, d))$
$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:
$(Y, \mathrm{pf}) \leftarrow \sigma$
If $(m \in S)$ then $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(0)$
$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$
If (vf) then $b' \leftarrow 1$ else $b' \leftarrow 0$
$\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(b')$

Figure 23: Left: Signature scheme DS_2. Right: Some adversaries for Theorem 8.5.

Proof of Theorem 8.5: Consider the games of Figure 25. Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2$ differ only in one line, the ones used, respectively, in these games, being lines 7,8,9, which change how the game decides whether or not the adversary's forgery attempt should let it win the game. We have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}_2}(\mathrm{A}_{\mathrm{ds}}) &= \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] \\ &= \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] + (\Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})]) \; .\end{aligned}$$

We claim to have designed the Figure 24 adversary $\mathrm{A}_{\mathrm{snd1}}$ so that

$$\Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}}) \; .$$

Next we have

$$\Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] = \Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})] + (\Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})]) \; .$$

Adversary $\mathrm{A}_{\mathrm{snd1}}$:

$cp \leftarrow\!\!\$\; \mathsf{CS.P}$; $K \leftarrow\!\!\$\; \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\!\!\$\; \mathsf{CS.D}$
$c \leftarrow \mathsf{CS.C}(cp, K, d)$; $\mathrm{crs} \leftarrow\!\!\$\; \mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi,TrCl_{CV}}}.\textsc{Init}$
$vk \leftarrow (\mathrm{crs}, cp, c)$; $\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:
Return vk

$\textsc{Sign}(m)$:
$Y \leftarrow \mathsf{F}(K, m)$
$\mathrm{pf} \leftarrow\!\!\$\; \mathsf{\Pi.P}(\mathrm{crs}, (cp, c, Y, m), (K, d))$
Return (Y, pf)

$\textsc{Fin}(m, \sigma)$:
$(Y, \mathrm{pf}) \leftarrow \sigma$
$\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi,TrCl_{CV}}}.\textsc{Vf}((cp, c, Y, m), \mathrm{pf})$
$\mathbf{G}^{\mathrm{snd1}}_{\mathsf{\Pi,TrCl_{CV}}}.\textsc{Fin}$

Adversary $\mathrm{A}_{\mathrm{hide}}$:

$cp \leftarrow\!\!\$\; \mathbf{G}^{\mathrm{hide}}_{\mathsf{CS}}.\textsc{Init}$; $K, K' \leftarrow\!\!\$\; \{0,1\}^{\mathsf{F.kl}}$
$c \leftarrow\!\!\$\; \mathbf{G}^{\mathrm{hide}}_{\mathsf{CS}}.\mathrm{CMT}(K', K)$; $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\; \mathsf{S.C}$
$vk \leftarrow (\mathrm{crs}, cp, c)$; $\mathrm{A}_{\mathrm{ds}}^{\textsc{Init},\textsc{Sign},\textsc{Fin}}$

$\textsc{Init}$:
Return vk

$\textsc{Sign}(m)$:
$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\!\!\$\; \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (cp, c, Y, m))$
$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:
$(Y, \mathrm{pf}) \leftarrow \sigma$
If $(m \in S)$ then $\mathbf{G}^{\mathrm{hide}}_{\mathsf{CS}}.\textsc{Fin}(0)$
$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$
If (vf) then $b' \leftarrow 1$ else $b' \leftarrow 0$
$\mathbf{G}^{\mathrm{hide}}_{\mathsf{CS}}.\textsc{Fin}(b')$

Figure 24: More adversaries for Theorem 8.5.

The assumption that CS is perfectly binding implies that

$$\Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{ds}})] = \Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})] \ .$$

Next we have

$$\Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})] = \Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})] + (\Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})]) \ .$$

We claim to have designed the Figure 23 adversary A_{zk} so that

$$\Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{zk}}_{\mathsf{\Pi,CV,S}}(\mathrm{A}_{\mathrm{zk}}) \ .$$

Next we have

$$\Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})] = \Pr[\mathrm{G}_4(\mathrm{A}_{\mathrm{ds}})] + (\Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_4(\mathrm{A}_{\mathrm{ds}})]) \ .$$

We claim to have designed the Figure 24 adversary $\mathrm{A}_{\mathrm{hide}}$ so that

$$\Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{ds}})] - \Pr[\mathrm{G}_4(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{hide}}_{\mathsf{CS}}(\mathrm{A}_{\mathrm{hide}}) \ .$$

Finally we claim to have designed the Figure 23 adversary A_{uf} so that

$$\Pr[\mathrm{G}_4(\mathrm{A}_{\mathrm{ds}})] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{F}}(\mathrm{A}_{\mathrm{uf}}) \ .$$

We omit the discussions of the adversaries and claims above due to lack of time. They will be discussed in class. ▮

8.4 Construction from ZK+XT1

When we refer to ZK+XT1, we mean that there is a single simulator S specifying $\mathsf{S.C}, \mathsf{S.P}$ and $\mathsf{S.X}$, with ZK holding with $\mathsf{S.C}, \mathsf{S.P}$, and XT1 holding with $\mathsf{S.C}, \mathsf{S.X}$.

Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2$

$\textsc{Init}()$:

$cp \leftarrow\!\!\$\ \mathsf{CS.P}$; $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\!\!\$\ \mathsf{CS.D}$; $c \leftarrow \mathsf{CS.C}(cp, K, d)$

$\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$; $vk \leftarrow (\mathrm{crs}, cp, c)$; Return vk

$\textsc{Sign}(m)$:

$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\!\!\$\ \mathsf{\Pi.P}(\mathrm{crs}, (cp, c, Y, m), (K, d))$

$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:

$(Y, \mathrm{pf}) \leftarrow \sigma$

If $(m \in S)$ then return false

$\mathrm{vf} \leftarrow \mathsf{\Pi.V}(\mathrm{crs}, (cp, c, Y, m), \mathrm{pf})$ // Game G_0

$\mathrm{vf} \leftarrow \exists (K', d') : \mathsf{CS.C}(cp, K', d') = c$ and $(\mathsf{F}(K', m) = Y)$ // Game G_1

$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$ // Game G_2

Return vf

Games $\mathrm{G}_3, \mathrm{G}_4$

$\textsc{Init}()$:

$cp \leftarrow\!\!\$\ \mathsf{CS.P}$; $K, K' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\!\!\$\ \mathsf{CS.D}$

$c \leftarrow \mathsf{CS.C}(cp, K, d)$ // Game G_3

$c \leftarrow \mathsf{CS.C}(cp, K', d)$ // Game G_4

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$; $vk \leftarrow (\mathrm{crs}, cp, c)$; Return vk

$\textsc{Sign}(m)$:

$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (cp, c, Y, m))$

$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:

$(Y, \mathrm{pf}) \leftarrow \sigma$

If $(m \in S)$ then return false

$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$; Return vf

Figure 25: Games for proof of Theorem 8.5.

Exercise 8.6 *Consider again the signature scheme* DS_2 *of Figure 23. You are to state and prove a result showing, again, its uf security, with the following changes in the assumptions: relax the assumption on* CS *from perfect binding to computational, and change the assumption on* $\mathsf{\Pi}$ *from ZK+SND1 to ZK+XT1. (As usual ZK+XT1 means there is a single simulator* S*, specifying* $\mathsf{S.C}, \mathsf{S.P}, \mathsf{S.X}$*, across the two.) Other assumptions stay the same:* F *is uf-secure,* CS *is computationally hiding. As a hint, Figure 26 shows games you may use. At some point you should be able to reuse claims and adversaries from the proof of Theorem 8.5, and you can refer to these rather than repeat them.*

$\underline{\text{Games } G_0, G_1, G_2}$

$\textsc{Init}()$:

$cp \leftarrow\!\!\$\ \mathsf{CS.P}$; $K \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\!\!\$\ \mathsf{CS.D}$; $c \leftarrow \mathsf{CS.C}(cp, K, d)$
$\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$ // Game G_0
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$ // Game G_1
$vk \leftarrow (\mathrm{crs}, cp, c)$; Return vk

$\textsc{Sign}(m)$:

$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, (cp, c, Y, m), (K, d))$
$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:

$(Y, \mathrm{pf}) \leftarrow \sigma$
If $(m \in S)$ then return false
$\mathrm{vf} \leftarrow \Pi.\mathsf{V}(\mathrm{crs}, (cp, c, Y, m), \mathrm{pf})$ // Games G_0, G_1
$(K', d') \leftarrow\!\!\$\ \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, (cp, c, Y, m), \mathrm{pf})$ // Game G_2
$\mathrm{vf} \leftarrow (\ \mathsf{CS.C}(cp, K', d') = c$ and $(\mathsf{F}(K', m) = Y)\ )$ // Game G_2
$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$ // Game G_3
Return vf

$\underline{\text{Games } G_4, G_5}$

$\textsc{Init}()$:

$cp \leftarrow\!\!\$\ \mathsf{CS.P}$; $K, K' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{F.kl}}$; $d \leftarrow\!\!\$\ \mathsf{CS.D}$
$c \leftarrow \mathsf{CS.C}(cp, K, d)$ // Game G_3
$c \leftarrow \mathsf{CS.C}(cp, K', d)$ // Game G_4
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$; $vk \leftarrow (\mathrm{crs}, cp, c)$; Return vk

$\textsc{Sign}(m)$:

$Y \leftarrow \mathsf{F}(K, m)$; $\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (cp, c, Y, m))$
$\sigma \leftarrow (Y, \mathrm{pf})$; $S \leftarrow S \cup \{m\}$; Return σ

$\textsc{Fin}(m, \sigma)$:

$(Y, \mathrm{pf}) \leftarrow \sigma$
If $(m \in S)$ then return false
$\mathrm{vf} \leftarrow (\mathsf{F}(K, m) = Y)$; Return vf

Figure 26: Games for Exercise 8.6.

9 Asymmetric encryption from NIZKs

We explore different ways to build an IND-CCA asymmetric encryption scheme from an IND-CPA asymmetric encryption scheme and a NIZK proof system.

9.1 Encryption definitions

An asymmetric (also called public key) encryption scheme AE specifies algorithms for key-generation, encryption and decryption, as follows. Via $(dk, ek) \leftarrow\!\!\$\ \mathsf{AE.K}$, the key-generation algorithm returns

Game $\mathbf{G}^{\text{ind}}_{\mathsf{AE}}$

INIT:
$(dk, ek) \leftarrow\$ \mathsf{AE.K}$; $b \leftarrow\$ \{0,1\}$;Return ek

ENC(m):
If $(b = 1)$ then $c \leftarrow\$ \mathsf{AE.E}(ek, m)$ else $c \leftarrow\$ \mathsf{AE.E}(ek, 0^{|m|})$
$Q \leftarrow Q \cup \{c\}$; Return c

DEC(c):
If $c \in Q$ then return $\perp$
$m \leftarrow \mathsf{AE.D}(dk, ek, c)$; Return m

FIN(b'):
Return $(b' = b)$

Figure 27: Game defining IND security of the public-key encryption scheme AE.

a public encryption key ek and a secret decryption key dk. Via $c \leftarrow \mathsf{AE.E}(ek, m; r)$, the encryption algorithm takes the encryption key ek, a message m, and randomness $r \in \{0,1\}^{\mathsf{AE.rl}}$ and returns a ciphertext c, where $\mathsf{AE.rl} \in \mathbb{N}$ is called the randomness length of AE. Thus $c \leftarrow\$ \mathsf{AE.E}(ek, m)$ means we let $r \leftarrow\$ \{0,1\}^{\mathsf{AE.rl}}$ and then let $c \leftarrow \mathsf{AE.E}(ek, m; r)$. Via $m \leftarrow \mathsf{AE.D}(ek, dk, c)$, the decryption algorithm takes the encryption and decryption keys and a ciphertext c to return either a message in $\{0,1\}^*$ or the special symbol $\perp$ indicating failure. Correctness requires that $\mathsf{AE.D}(dk, ek, \mathsf{AE.E}(ek, m; r)) = m$ for all $(dk, ek) \in [\mathsf{AE.K}]$, all $m \in \{0,1\}^*$ and all $r \in \{0,1\}^{\mathsf{AE.rl}}$.

We will capture the IND-CPA, IND-CCA1 and IND-CCA2 notions of security for AE via the single game $\mathbf{G}^{\text{ind}}_{\mathsf{AE}}$ shown in Figure 27. If A is an ind-adversary we let $\mathbf{Adv}^{\text{ind}}_{\mathsf{AE}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\text{ind}}_{\mathsf{AE}}(\mathrm{A})] - 1$ be its ind-advantage. Now we define different classes (sets) of adversaries:

- $\mathcal{A}_{\text{cca2}}$ is the set of all ind-adversaries A making at most one ENC query. Security relative to these is IND-CCA2.
- $\mathcal{A}_{\text{cca1}} \subseteq \mathcal{A}_{\text{cca2}}$ is the set of all $\mathrm{A} \in \mathcal{A}_{\text{cca2}}$ that make their DEC queries prior to their ENC query. (That is, once they have made an ENC query, they do not make any DEC queries.) Security relative to these is IND-CCA1.
- $\mathcal{A}_{\text{cpa}} \subseteq \mathcal{A}_{\text{cca1}}$ is the set of all $\mathrm{A} \in \mathcal{A}_{\text{cca2}}$ that make zero DEC queries. Security relative to these is IND-CPA.

9.2 The NY Encryption Scheme

Let AE be an asymmetric encryption scheme. Let CV be the claim validator of Figure 28. Let Π be a proof system. Associate to AE, Π the asymmetric encryption scheme $\overline{\mathsf{AE}}$ whose algorithms are shown in Figure 28.

9.3 IND-CCA1 security of the NY scheme

Theorem 9.1 *Let* AE *be an asymmetric encryption scheme. Let* CV *be the claim validator of Figure 28. Let* Π *be a proof system that satisfies SND1 and ZK for* CV*. Let* $\overline{\mathsf{AE}}$ *be the asymmetric*

Claim validator $\mathsf{CV}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), (m, r_0, r_1))$:

$d_0 \leftarrow (c_0 = \mathsf{AE.E}(ek_0, m; r_0))$; $d_1 \leftarrow (c_1 = \mathsf{AE.E}(ek_1, m; r_1))$; Return (d_0 and d_1)

Key-generation algorithm $\overline{\mathsf{AE}}.\mathsf{K}$:

$(dk_0, ek_0) \leftarrow\!\!\$\ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\!\!\$\ \mathsf{AE.K}$; $\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$

$\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$; $\overline{dk} \leftarrow (dk_0, dk_1)$; Return $(\overline{dk}, \overline{ek})$

Encryption algorithm $\overline{\mathsf{AE}}.\mathsf{E}((ek_0, ek_1, \mathrm{crs}), m)$:

$r_0, r_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{AE.rl}}$; $c_0 \leftarrow \mathsf{AE.E}(ek_0, m; r_0)$; $c_1 \leftarrow \mathsf{AE.E}(ek_1, m; r_1)$

$\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), (m, r_0, r_1))$; Return (c_0, c_1, pf)

Decryption algorithm $\overline{\mathsf{AE}}.\mathsf{D}((dk_0, dk_1), (ek_0, ek_1, \mathrm{crs}), (c_0, c_1, \mathrm{pf}))$:

If $\Pi.\mathsf{V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$ then return $\perp$

Return $m \leftarrow \mathsf{AE.D}(dk_0, ek_0, c_0)$

Figure 28: NY encryption scheme.

---

encryption scheme whose algorithms are shown in Figure 28. Let $\mathrm{A}_{\mathrm{cca1}} \in \mathcal{A}_{\mathrm{cca1}}$ *be an adversary making* q *queries to its* DEC *oracle. Then we can construct an adversary*$\mathrm{A}_{\mathrm{cpa}} \in \mathcal{A}_{\mathrm{cpa}}$*, and adversaries* A_{zk} *and* $\mathrm{A}_{\mathrm{snd1}}$ *(shown explicitly in Figure 31) such that*

$$\mathbf{Adv}^{\mathrm{ind}}_{\overline{\mathsf{AE}}}(\mathrm{A}_{\mathrm{cca1}}) \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}}) + \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk}}) + \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}}(\mathrm{A}_{\mathrm{cpa}}) \ . \tag{10}$$

The running time of $\mathrm{A}_{\mathrm{cpa}}$ *is that of* $\mathrm{A}_{\mathrm{cca1}}$ *plus* $\mathcal{O}(q \cdot \mathrm{T}(\mathsf{S.P}) + \mathrm{T}(\mathsf{S.C}) + \mathrm{T}(\mathsf{CS.C}) + \mathrm{T}(\mathsf{CS.P}))$*. The running time of* A_{zk} *is that of* $\mathrm{A}_{\mathrm{cca1}}$ *plus ...*

Proof of Theorem 9.1: We assume that, in game $\mathbf{G}^{\mathrm{ind}}_{\overline{\mathsf{AE}}}$ of Figure 27, adversary $\mathrm{A}_{\mathrm{cca1}}$ never queries DEC(c) for $c \in Q$, so lines 3,4 can be removed. Now consider the games of Figure 29, and let $p_i = \Pr[\mathrm{G}_i(\mathrm{A}_{\mathrm{cca1}})]$ for $i \in [0..4]$. Then

$$\begin{aligned}\Pr[\mathbf{G}^{\mathrm{ind}}_{\overline{\mathsf{AE}}}(\mathrm{A}_{\mathrm{cca1}})] &= p_0 \\ &= (p_0 - p_1) + (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4) + p_4 \ .\end{aligned}$$

Thus

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind}}_{\overline{\mathsf{AE}}}(\mathrm{A}_{\mathrm{cca1}}) &= 2p_0 - 1 \\ &= 2(p_0 - p_1) + 2(p_1 - p_2) + 2(p_2 - p_3) + 2(p_3 - p_4) + (2p_4 - 1) \ .\end{aligned}$$

Game G_1 switches to simulated CRS (line 4 instead of 3) and proofs (line 13 instead of 12), these being the only changes. We can design a zk-adversary $\mathrm{A}_{\mathrm{zk},1}$ (so denoted because later we will design another zk-adversary $\mathrm{A}_{\mathrm{zk},2}$) such that

$$p_0 - p_1 \leq \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk},1}) \ .$$

The design is quite easy. Adversary $\mathrm{A}_{\mathrm{zk},1}$ mimics lines 1,2, gets crs from its own INIT oracle, then continues to create $\overline{dk}$ as per line 6. It now runs $\mathrm{A}_{\mathrm{cca1}}$, providing $\overline{dk}$ in response to the latter's INIT query. In answering the (assumed single) ENC query, it mimics the game, except that it calls its ownn PF oracle in place of lines 12/13 (it has the witness) to get pf. Queries to DEC are easily answered since it knows dk_0. Obtaining b' from $\mathrm{A}_{\mathrm{cca1}}$'s FIN query, its own FIN query is 1 if $b' = b$

<u>Games G_0–G_4</u>

INIT:
1 $b \leftarrow\!\!\$ \{0,1\}$
2 $(dk_0, ek_0) \leftarrow\!\!\$ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\!\!\$ \mathsf{AE.K}$
3 $\mathrm{crs} \leftarrow\!\!\$ \mathsf{\Pi.C}$ // Game G_0
4 $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$ \mathsf{S.C}$ // Games G_1–G_4
5 $\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$
6 Return $\overline{ek}$

ENC(m):
7 $r_0, r_1 \leftarrow\!\!\$ \{0,1\}^{\mathsf{AE.rl}}$; $m_1 \leftarrow m$; $m_0 \leftarrow 0^{|m|}$
8 $c_0 \leftarrow \mathsf{AE.E}(ek_0, m_b; r_0)$ // Games G_0–G_3
9 $c_0 \leftarrow \mathsf{AE.E}(ek_0, m_0; r_0)$ // Game G_4
10 $c_1 \leftarrow \mathsf{AE.E}(ek_1, m_b; r_1)$ // Games $\mathrm{G}_0, \mathrm{G}_1$
11 $c_1 \leftarrow \mathsf{AE.E}(ek_1, m_0; r_1)$ // Game G_2–G_4
12 $\mathrm{pf} \leftarrow\!\!\$ \mathsf{\Pi.P}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), (m, r_0, r_1))$ // Game G_0
13 $\mathrm{pf} \leftarrow\!\!\$ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (ek_0, ek_1, c_0, c_1))$ // Games G_1–G_4
14 Return (c_0, c_1, pf)

DEC((c_0, c_1, pf)):
15 If $\mathsf{\Pi.V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$ then return $\perp$
16 $m \leftarrow \mathsf{AE.D}(dk_0, ek_0, c_0)$ // Games G_0–G_2
17 $m \leftarrow \mathsf{AE.D}(dk_1, ek_1, c_1)$ // Games $\mathrm{G}_3, \mathrm{G}_4$
18 Return m

FIN(b'):
19 Return $(b' = b)$

Figure 29: Games for proof of Theorem 9.1.

and 0 otherwise, where b is the bit it itself chose in accordance with line 1.

The only change for game G_2 is that line 10 is replaced by line 11, where c_1 is created by encrypting m_0 rather than m_b. We design ind-adversary $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}}(\mathrm{A}_{\mathrm{ind},1}) \in \mathcal{A}_{\mathrm{cpa}}$ (it is a cpa adversary, meaning it makes no queries to its own DEC, and is again so denoted because there will be another cpa adversary below) such that

$$p_1 - p_2 \le \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}}(\mathrm{A}_{\mathrm{ind},1}) \; .$$

Adversary $\mathrm{A}_{\mathrm{ind},1}$ gets ek_1 from its INIT oracle, but otherwise follows lines 1–5 of G_1 to create $\overline{dk}$, and then runs $\mathrm{A}_{\mathrm{cca1}}$, providing $\overline{dk}$ in response to the latter's INIT query. Since it knows dk_0, it can answers DEC queries as per line 16. For the ENC query, it sends m_b to its own ENC to get back c_1, and itself generates pf as per line 13. Getting b' from $\mathrm{A}_{\mathrm{cca1}}$, it calls its own FIN with 1 if $b' = b$ and 0 otherwise.

The only change for G_3 is that line 16 is replaced by line 17, so that decryption is performed with dk_1 rather than with dk_0. Bounding $p_2 - p_3$ is the interesting and harder element of the proof, and we postpone it. Continuing, the only change for G_4 is that line 8 is replaced by line 9, where c_0 is

Games $\boxed{\mathrm{G}_5}$, G_6

INIT:

1 $b \leftarrow\!\!\$\ \{0,1\}$
2 $(dk_0, ek_0) \leftarrow\!\!\$\ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\!\!\$\ \mathsf{AE.K}$
3 $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$
4 $\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$
5 Return $\overline{ek}$

ENC(m):

6 $r_0, r_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{AE.rl}}$; $m_1 \leftarrow m$; $m_0 \leftarrow 0^{|m|}$
7 $c_0 \leftarrow \mathsf{AE.E}(ek_0, m_b; r_0)$
8 $c_1 \leftarrow \mathsf{AE.E}(ek_1, m_0; r_1)$
9 $\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, (ek_0, ek_1, c_0, c_1))$
10 Return (c_0, c_1, pf)

DEC((c_0, c_1, pf)):

11 If $\mathsf{\Pi.V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$
12   then return $\perp$
13 $m \leftarrow \mathsf{AE.D}(dk_1, ek_1, c_1)$
14 $m' \leftarrow \mathsf{AE.D}(dk_0, ek_0, c_0)$
15 If $(m' \neq m)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{m \leftarrow m'}$
16 Return m

FIN(b'):

17 Return $(b' = b)$

Games G_7, G_8

INIT:

1 $(dk_0, ek_0) \leftarrow\!\!\$\ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\!\!\$\ \mathsf{AE.K}$
2 $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$ // Game G_7
3 $\mathrm{crs} \leftarrow\!\!\$\ \mathsf{\Pi.C}$ // Game G_8
4 $\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$
5 Return $\overline{ek}$

ENC(m):

6 Return $\perp$

DEC((c_0, c_1, pf)):

7 If $\mathsf{\Pi.V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$
8   then return $\perp$
9 $m \leftarrow \mathsf{AE.D}(dk_1, ek_1, c_1)$
10 $m' \leftarrow \mathsf{AE.D}(dk_0, ek_0, c_0)$
11 If $(m' \neq m)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
12 Return m

FIN(b'):

13 Return bad

Figure 30: Games for proof of Theorem 9.1.

created by encrypting m_0 rather than m_b. As above, we can design $\mathrm{A} \in \mathcal{A}_{\mathrm{cpa}}$ such that

$$p_3 - p_4 \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{AE}}(\mathrm{A}_{\mathrm{ind},2}) \ .$$

Adversary $\mathrm{A}_{\mathrm{ind},2}$ gets ek_0 from its INIT oracle, and then proceeds analogously to before. The simulation of DEC by $\mathrm{A}_{\mathrm{ind},2}$ is possible because this now uses dk_1 at line 17, which $\mathrm{A}_{\mathrm{ind},2}$ knows.

In G_4, both c_0 and c_1 are encryptions of m_0, and the bit b is not used, so

$$p_4 = \frac{1}{2} \ .$$

It remains to bound $p_2 - p_3$. Consider games $\mathrm{G}_5, \mathrm{G}_6$ in Figure 30. We have

$$p_2 = \Pr[\mathrm{G}_5(\mathrm{A}_{\mathrm{cca1}})]$$
$$p_3 = \Pr[\mathrm{G}_6(\mathrm{A}_{\mathrm{cca1}})] \ .$$

Since these games are identical-until-bad we have

$$p_2 - p_3 \leq \Pr[\mathrm{G}_6(\mathrm{A}_{\mathrm{cca1}}) \text{ sets } \mathsf{bad}] \ .$$

Now consider games $\mathrm{G}_7, \mathrm{G}_8$ in the same Figure. Game G_7 returns bad as its output and replies $\perp$ to the ENC query. By the assumption that $\mathrm{A}_{\mathrm{cca1}}$ is in $\mathcal{A}_{\mathrm{cca1}}$ (this is where we use the assumption!) all its DEC queries precede its ENC query, so if bad is set then it is set prior to the ENC query, Adversary $\mathrm{A}_{\mathrm{snd1}}$:

$(dk_0, ek_0) \leftarrow\$ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\$ \mathsf{AE.K}$
$\mathrm{crs} \leftarrow\$ \mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Init}$
$\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$
$\mathrm{A}_{\mathrm{cca1}}^{\textsc{Init},\textsc{Enc},\textsc{Dec},\textsc{Fin}}$

$\textsc{Init}$:
Return $\overline{ek}$

$\textsc{Enc}(m)$:
Return $\perp$

$\textsc{Dec}((c_0, c_1, \mathrm{pf}))$:
If $\Pi.\mathsf{V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$
  then return $\perp$
$\mathrm{vf} \leftarrow \mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Vf}((ek_0, ek_1, c_0, c_1))$
$m \leftarrow \mathsf{AE.D}(dk_1, ek_1, c_1)$
Return m

$\textsc{Fin}(b')$:
$\mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}.\textsc{Fin}$

Adversary $\mathrm{A}_{\mathrm{zk},2}$:

$(dk_0, ek_0) \leftarrow\$ \mathsf{AE.K}$; $(dk_1, ek_1) \leftarrow\$ \mathsf{AE.K}$
$\mathrm{crs} \leftarrow\$ \mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Init}$
$\overline{ek} \leftarrow (ek_0, ek_1, \mathrm{crs})$
$\mathrm{A}_{\mathrm{cca1}}^{\textsc{Init},\textsc{Enc},\textsc{Dec},\textsc{Fin}}$

$\textsc{Init}$:
Return $\overline{ek}$

$\textsc{Enc}(m)$:
Return $\perp$

$\textsc{Dec}((c_0, c_1, \mathrm{pf}))$:
If $\Pi.\mathsf{V}(\mathrm{crs}, (ek_0, ek_1, c_0, c_1), \mathrm{pf}) = \mathsf{false}$
  then return $\perp$
$m \leftarrow \mathsf{AE.D}(dk_1, ek_1, c_1)$
$m' \leftarrow \mathsf{AE.D}(dk_0, ek_0, c_0)$
If $(m' \neq m)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
Return m

$\textsc{Fin}(b')$:
If bad then $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(0)$
Else $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}.\textsc{Fin}(1)$

Figure 31: More adversaries for Theorem 9.1.

meaning what is returned for the latter does not change the outcome of game G_7. Thus we have

$$\begin{aligned}\Pr[\mathrm{G}_6(\mathrm{A}_{\mathrm{cca1}}) \text{ sets } \mathsf{bad}] &= \Pr[\mathrm{G}_7(\mathrm{A}_{\mathrm{cca1}})] \\ &= \Pr[\mathrm{G}_8(\mathrm{A}_{\mathrm{cca1}})] + (\ \Pr[\mathrm{G}_7(\mathrm{A}_{\mathrm{cca1}})] - \Pr[\mathrm{G}_8(\mathrm{A}_{\mathrm{cca1}})]\ )\ .\end{aligned}$$

Game G_8 has switched the simulated crs at line 3 back to the real crs at line 4. We can now design snd1-adversary $\mathrm{A}_{\mathrm{snd1}}$ such that

$$\Pr[\mathrm{G}_8(\mathrm{A}_{\mathrm{cca1}})] \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TrCl}_{\mathsf{CV}}}(\mathrm{A}_{\mathrm{snd1}})\ .$$

Finally we can design zk-adversary $\mathrm{A}_{\mathrm{zk},2}$ such that

$$\Pr[\mathrm{G}_7(\mathrm{A}_{\mathrm{cca1}})] - \Pr[\mathrm{G}_8(\mathrm{A}_{\mathrm{cca1}})] \leq \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{zk},2})\ .$$

The two zk-adversaries can be combined into one whose advantage is the sum of the original ones, and similarly for the two ind-adversaries, so that the theorem statement needs to speak of just one adversary of any particular type.

A full proof should give proper descriptions of all the constructed adversaries and say what is their runtime, and the number of queries they make to their various oracles, but we are out of time. ∎

The interesting question here is, why would the above proof not work to show that the scheme is IND-CCA2? The issue comes up only in bounding $p_2 - p_3$ via games G_5–G_8. We would like to not return $\perp$ as in line 7 of $\mathrm{G}_7, \mathrm{G}_8$, instead responding as per G_6. So, define G_7' to be like G_7 except that replies to $\textsc{Enc}$ are as in G_6, and define G_8' to be like G_8 except that replies to $\textsc{Enc}$ are as in G_6 with the further change that $\Pi.\mathsf{P}$ is used to generate the proof at line 9 rather than $\mathsf{S.P}$. The snd1 adversary can in fact simulate the ENC replies of game G'_8, so we can bound $\Pr[G'_8(A_{cca1})]$. Can we build the zk-adversary $A_{zk,2}$ to bound $\Pr[G'_7(A_{cca1})] - \Pr[G'_8(A_{cca1})]$? At first glance, yes, because it can use its own PF oracle to compute the proof pf at line 9. But here we have a problem. Proofs submitted to PF must be of true claims, else the oracle returns $\perp$. But the claim here is a false one, because c_0, c_1 do *not* encrypt the same message.

10 Achieving simulation soundness

We want to build a proof system $\overline{\mathsf{\Pi}}$ that is ZK+SND3. It is done from a proof system $\mathsf{\Pi}$ satisfying some other, weaker conditions. The transform, due to Groth [Gro06], uses signature schemes.

A first question is what properties it is useful or appropriate to assume of $\mathsf{\Pi}$. We start with a definition. If S is a simulator, we say that $\mathsf{\Pi}$ and S are same-CRS if $\mathsf{\Pi.C}$ works as follows:

$$(\mathrm{crs}, \mathrm{td}) \leftarrow\$ \mathsf{S.C} \; ; \; \text{Return crs} \; .$$

Thus, the CRSs created by the CRS generators of the proof system and the simulator are the same (more precisely, identically distributed). We assume that, given a (any) claim validator CV, we can obtain $\mathsf{\Pi}, \mathsf{S}$ such that

(1) $\mathsf{\Pi}$ and S are same-CRS

(2) $\mathsf{\Pi}$ is XT1 for CV relative to S

(3) $\mathsf{\Pi}$ is WI for CV.

Why these choices? Because we want to leverage what we have, which is the GOS system of Theorem 7.7, and can get this from there. We set $\mathsf{\Pi}$ to the mode 1 proof system $\mathsf{\Pi}_1$ from that theorem. The reason this is WI is that $\mathsf{\Pi}_0$ is ZK, hence WI, and WI transfers to $\mathsf{\Pi}_1$. Of course $\mathsf{\Pi}_1$ is ZK too, but we will not need this.

Now assume we are given a target claim validator $\overline{\mathsf{CV}}$. Most likely it is for an **NP**-language. As auxiliary tools, we have a pair $\mathsf{DS}_1, \mathsf{DS}_2$ of signature schemes, the first UF-secure and the second SUF-secure. To these, we associate a claim validator CV as defined in Figure 32. By the above, we can assume we have a proof system $\mathsf{\Pi}$ and simulator S such that the three conditions above are true. We now build proof system $\overline{\mathsf{\Pi}}$ and simulator $\overline{\mathsf{S}}$ such that

(1) $\overline{\mathsf{\Pi}}$ is ZK for $\overline{\mathsf{CV}}$ relative to $\overline{\mathsf{S}}$

(2) $\overline{\mathsf{\Pi}}$ is SND3 for $\overline{\mathsf{CV}}$ relative to $\overline{\mathsf{S}}$.

The algorithms of $\overline{\mathsf{\Pi}}$, and those of $\overline{\mathsf{S}}$, are shown in Figure 32.

The idea can be seen from the definition of CV. Given crs, claim x and candidate witness w, claim validator CV returns true if one of two things is true: either (1) w is a valid witness for x relative to the given claim validator $\overline{\mathsf{CV}}$, or (2) w is a valid signature of message (x, vk_2) relative to verification key vk_1. The corresponding signing key sk_1 will be the trapdoor for simulator $\overline{\mathsf{S}}$.

Theorem 10.1 *Let $\mathsf{\Pi}$ be a proof system, and S a simulator, that are same-CRS. Let $\mathsf{DS}_1, \mathsf{DS}_2$ be signature schemes. Let $\overline{\mathsf{CV}}$ be a claim validator. Define claim validator CV, proof system $\overline{\mathsf{\Pi}}$ and simulator $\overline{\mathsf{S}}$ as per Figure 32. Then:*

1. *If $\mathsf{\Pi}$ is WI for CV then $\overline{\mathsf{\Pi}}$ is ZK for $\overline{\mathsf{CV}}$ relative to $\overline{\mathsf{S}}$: Let A_{zk} be an adversary making q queries to its PF oracle. Then we can construct an adversary A_{wi} (shown explicitly in Figure 33) such*

Claim validator $\mathsf{CV}(\mathrm{crs}, (x, vk_1, vk_2), w)$:

$d_1 \leftarrow (\overline{\mathsf{CV}}((\mathrm{crs}, vk_1), x, w)$
$d_2 \leftarrow (\mathsf{DS.V}(vk_1, (x, vk_2), w)$
Return $(d_1$ or $d_2)$

CRS generator $\overline{\Pi}.\mathsf{C}$:

$(sk_1, vk_1) \leftarrow\!\!\$\ \mathsf{DS}_1.\mathsf{K}$
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$
Return (crs, vk_1)

Proof generator $\overline{\Pi}.\mathsf{P}((\mathrm{crs}, vk_1), x, w)$:

$(sk_2, vk_2) \leftarrow\!\!\$\ \mathsf{DS}_2.\mathsf{K}$
$\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, (x, vk_1, vk_2), w)$
$\sigma_2 \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_2, vk_2, (x, \mathrm{pf}))$
Return $(\mathrm{pf}, vk_2, \sigma_2)$

Proof verifier $\overline{\Pi}.\mathsf{V}((\mathrm{crs}, vk_1), x, (\mathrm{pf}, vk_2, \sigma_2))$:

$\mathrm{vf}_1 \leftarrow \mathsf{DS}_2.\mathsf{V}(vk_2, (x, \mathrm{pf}), \sigma_2)$
$\mathrm{vf}_2 \leftarrow \Pi.\mathsf{V}(\mathrm{crs}, (x, vk_1, vk_2), \mathrm{pf})$
Return $(\mathrm{vf}_1$ and $\mathrm{vf}_2)$

Sim CRS generator $\overline{\mathsf{S}}.\mathsf{C}$:

$(sk_1, vk_1) \leftarrow\!\!\$\ \mathsf{DS}_1.\mathsf{K}$
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$
Return $((\mathrm{crs}, vk_1), (sk_1, \mathrm{td}))$

Sim proof generator $\overline{\mathsf{S}}.\mathsf{P}((\mathrm{crs}, vk_1), (sk_1, \mathrm{td}), x)$:

$(sk_2, vk_2) \leftarrow\!\!\$\ \mathsf{DS}_2.\mathsf{K}$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{DS}_1.\mathsf{S}(sk_1, vk_1, (x, vk_2))$
$\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}(\mathrm{crs}, (x, vk_1, vk_2), \sigma_1)$
$\sigma_2 \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_2, vk_2, (x, \mathrm{pf}))$
Return $(\mathrm{pf}, vk_2, \sigma_2)$

Figure 32: Proof system $\overline{\Pi}$ and simulator $\overline{\mathsf{S}}$ for Theorem 10.1.

---

that

$$\mathbf{Adv}^{\mathrm{zk}}_{\overline{\Pi},\overline{\mathsf{CV}},\overline{\mathsf{S}}}(\mathrm{A}_{\mathrm{zk}}) \leq \mathbf{Adv}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}_{\mathrm{wi}}) \ . \tag{11}$$

Adversary A_{wi} makes q queries to its Pf *oracle and its running time is about that of A_{zk}.*

2. *If Π is XT1 for CV, signature scheme DS_1 is UF and signature scheme DS_2 is SUF then $\overline{\Pi}$ is SND3 for $\overline{\mathsf{CV}}$ relative to $\overline{\mathsf{S}}$: Let $\mathrm{A}_{\mathrm{snd3}}$ be an adversary making q queries to its* Pf *oracle and one query to its* Vf *oracle. Then we can construct adversaries $\mathrm{A}_{\mathrm{xt1}}, \mathrm{A}_{\mathrm{ds1}}, \mathrm{A}_{\mathrm{ds2}}$ (shown explicitly in Figure 34) such that*

$$\mathbf{Adv}^{\mathrm{snd3}}_{\overline{\Pi},\overline{\mathsf{CV}},\overline{\mathsf{S}}}(\mathrm{A}_{\mathrm{snd3}}) \leq \mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\mathrm{xt1}}) + \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}_1}(\mathrm{A}_{\mathrm{ds1}}) + q \cdot \mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}_2}(\mathrm{A}_{\mathrm{ds2}}) \ . \tag{12}$$

Adversary $\mathrm{A}_{\mathrm{xt1}}$ makes 1 query to its Ex *oracle. Adversary $\mathrm{A}_{\mathrm{ds1}}$ makes q queries to its* Sign *oracle. Adversary $\mathrm{A}_{\mathrm{ds1}}$ makes 1 query to its* Sign *oracle. The running time of these adversaries is about that of $\mathrm{A}_{\mathrm{snd3}}$.*

Proof of Theorem 10.1: Adversary A_{wi} for part **1** is shown in Figure 33. For part **2**, consider the games of Figure 33. They differ only in their Fin procedures. We have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{snd3}}_{\overline{\Pi},\overline{\mathsf{CV}},\overline{\mathsf{S}}}(\mathrm{A}_{\mathrm{snd3}}) &= \Pr[\mathrm{G}_0(\mathrm{A}_{\mathrm{snd3}})] \\ &\leq \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd3}})] + \Pr[\mathrm{G}_2(\mathrm{A}_{\mathrm{snd3}})] + \Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{snd3}})] \ .\end{aligned}$$

Adversary $\mathrm{A}_{\mathrm{ds2}}$ is shown in Figure 34. We have

$$\mathbf{Adv}^{\mathrm{suf}}_{\mathsf{DS}_2}(\mathrm{A}_{\mathrm{ds2}}) \geq \frac{1}{q} \cdot \Pr[\mathrm{G}_1(\mathrm{A}_{\mathrm{snd3}})] \ .$$

Games G_0–G_3

$\textsc{Init}()$:

1 $(sk_1, vk_1) \leftarrow\$ \mathsf{DS}_1.\mathsf{K}$

2 $(\mathrm{crs}, \mathrm{td}) \leftarrow\$ \mathsf{S.C}$

3 Return (crs, vk_1)

$\textsc{Pf}(x)$:

4 $(sk_2, vk_2) \leftarrow\$ \mathsf{DS}_2.\mathsf{K}$

5 $\sigma_1 \leftarrow\$ \mathsf{DS}_1.\mathsf{S}(sk_1, vk_1, (x, vk_2))$

6 $\mathrm{pf} \leftarrow\$ \Pi.\mathsf{P}(\mathrm{crs}, (x, vk_1, vk_2), \sigma_1)$

7 $\sigma_2 \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_2, vk_2, (x, \mathrm{pf}))$

8 $Q \leftarrow Q \cup \{(x, (\mathrm{pf}, vk_2, \sigma_2))\}$

9 $S \leftarrow S \cup \{(x, vk_2)\}$

10 Return $(\mathrm{pf}, vk_2, \sigma_2)$

$\textsc{Vf}(x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*))$:

11 If $((x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*)) \in Q)$ then return false

12 If $(x^* \in \mathsf{TrCl}_{\overline{\mathsf{CV}}}((\mathrm{crs}, vk_1)))$ then return false

13 $\mathrm{vf}_1 \leftarrow \mathsf{DS}_2.\mathsf{V}(vk_2^*, (x^*, \mathrm{pf}^*), \sigma_2^*)$

14 $\mathrm{vf}_2 \leftarrow \Pi.\mathsf{V}(\mathrm{crs}, (x^*, vk_1, vk_2^*), \mathrm{pf}^*)$

15 If $(\mathrm{vf}_1$ and $\mathrm{vf}_2)$ then $\mathsf{win} \leftarrow \mathsf{true}$

16 $\sigma_1^* \leftarrow\$ \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, (x^*, vk_1, vk_2^*), \mathrm{pf}^*)$

17 $u_1 \leftarrow (\ (x^*, vk_2^*) \in S\ )$

18 $u_2 \leftarrow \mathsf{DS.V}(vk_1, (x^*, vk_2), \sigma_1^*)$

19 Return $(\mathrm{vf}_1$ and $\mathrm{vf}_2)$

$\textsc{Fin}()$: // Game G_0

20 Return win

$\textsc{Fin}()$: // Game G_1

21 Return $(\mathsf{win}$ and $u_1)$

$\textsc{Fin}()$: // Game G_2

22 Return $(\mathsf{win}$ and $\neg u_1$ and $\neg u_2)$

$\textsc{Fin}()$: // Game G_3

23 Return $(\mathsf{win}$ and $\neg u_1$ and $u_2)$

Adversary A_{wi}:

1 $(sk_1, vk_1) \leftarrow\$ \mathsf{DS}_1.\mathsf{K}$; $\mathrm{crs} \leftarrow\$ \mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}.\textsc{Init}$

2 $A_{\mathrm{zk}}^{\textsc{Init},\textsc{Pf},\textsc{Fin}}$

$\textsc{Init}$:

3 Return crs

$\textsc{Pf}(x, w)$:

4 If $\overline{\mathsf{CV}}((\mathrm{crs}, vk_1), x, w) = \mathsf{false}$ then return $\perp$

5 $w_1 \leftarrow w$; $(sk_2, vk_2) \leftarrow\$ \mathsf{DS}_2.\mathsf{K}$

6 $w_0 \leftarrow\$ \mathsf{DS}_1.\mathsf{S}(sk_1, vk_1, (x, vk_2))$

7 $\mathrm{pf} \leftarrow\$ \mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}.\textsc{Pf}((x, vk_1, vk_2), w_0, w_1)$

8 Return pf

$\textsc{Fin}(b')$:

9 $\mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}.\textsc{Fin}(b')$

Figure 33: WI-adversary for part **1** of Theorem 10.1, and games for part **2** of same.

Let us try to justify the above. Assume $vk_2^* = vk_{2,g}$, which happens with probability $1/q$ since $(x^*, vk_2^*) \in S$ as per lines 17,21 of the games. According to line 21, win is true, so as per line 15, $\mathrm{vf}_1 = \mathsf{true}$. Now the only reason A_{ds2} could fail to win its game is that $(x^*, \mathrm{pf}^*, \sigma_2^*) = (x_g, \mathrm{pf}_g, \sigma_{2,g})$. But if so, $(x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*))$ would be in the set Q, contradicting line 11 of the game.

One question above may be why A_{ds2} returns true in response to the $\textsc{Vf}$ query, and why this is ok. The answer is that it cannot efficiently perform the test at line 12 of the game, so cannot faithfully simulate the oracle. However, we have assumed only one $\textsc{Vf}$ query, and following it, A_{snd3} must make its mandatory $\textsc{Fin}$ query. What is returned to the $\textsc{Vf}$ query thus does not matter; all our adversary needs is to record the query itself. This will recur in what follows. (In retrospect however this leads me to think it is best to change the definitions of the SND1,SND2,SND3 games to not return the result of the language membership test to the adversary, using it instead only in deciding whether or not to set win. Otherwise, things get tougher when multiple $\textsc{Vf}$ queris are allowed, at least if we want good bounds.)

Adversary A_{xt1} is shown in Figure 34. We have

$$\Pr[G_2(A_{\mathrm{snd3}})] \leq \mathbf{Adv}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(A_{\mathrm{xt1}}) .$$

Let us try to justify the above. Line 22 says $\mathsf{win} = \mathsf{true}$, so $\mathrm{vf}_2 = \mathsf{true}$ at line 14, meaning the proof

Adversary $\mathrm{A}_{\mathrm{ds2}}$:

$(sk_1, vk_1) \leftarrow\!\$\; \mathsf{DS}_1.\mathsf{K}$; $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\$\; \mathsf{S.C}$
$g \leftarrow\!\$\; [1..q]$; $vk_{2,g} \leftarrow\!\$\; \mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}_2}.\textsc{Init}$
$\mathrm{A}_{\mathrm{snd3}}^{\textsc{Init},\textsc{Pf},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}$:
Return (crs, vk_1)

$\textsc{Pf}(x)$:
$i \leftarrow i+1$; $x_i \leftarrow x$
If $(i \neq g)$ then $(sk_{2,i}, vk_{2,i}) \leftarrow\!\$\; \mathsf{DS}_2.\mathsf{K}$
$\sigma_1 \leftarrow\!\$\; \mathsf{DS}_1.\mathsf{S}(sk_1, vk_1, (x_i, vk_{2,i}))$
$\mathrm{pf}_i \leftarrow\!\$\; \Pi.\mathsf{P}(\mathrm{crs}, (x_i, vk_1, vk_{2,i}), \sigma_1)$
If $(i = g)$ then $\sigma_{2,i} \leftarrow \mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}_2}.\textsc{Sign}((x_i, \mathrm{pf}_i))$
Else $\sigma_{2,i} \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_{2,i}, vk_{2,i}, (x_i, \mathrm{pf}_i))$
Return pf_i

$\textsc{Vf}(x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*))$:
Return true

$\textsc{Fin}()$:
$\mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}_2}.\textsc{Fin}((x^*, \mathrm{pf}^*), \sigma_2^*)$

Adversary $\mathrm{A}_{\mathrm{xt1}}$:

$(sk_1, vk_1) \leftarrow\!\$\; \mathsf{DS}_1.\mathsf{K}$; $\mathrm{crs} \leftarrow\!\$\; \mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV}}.\textsc{Init}$; $\mathrm{A}_{\mathrm{snd3}}^{\textsc{Init},\textsc{Pf},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}$:
Return (crs, vk_1)

$\textsc{Pf}(x)$:
$(sk_2, vk_2) \leftarrow\!\$\; \mathsf{DS}_2.\mathsf{K}$; $\sigma_1 \leftarrow\!\$\; \mathsf{DS}_1.\mathsf{S}(sk_1, vk_1, (x, vk_2))$
$\mathrm{pf} \leftarrow\!\$\; \Pi.\mathsf{P}(\mathrm{crs}, (x, vk_1, vk_2), \sigma_1)$
$\sigma_2 \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_2, vk_2, (x, \mathrm{pf}))$; Return pf

$\textsc{Vf}(x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*))$:
$d^* \leftarrow\!\$\; \mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV}}.\mathrm{Ex}((x^*, vk_1, vk_2^*), \mathrm{pf}^*)$; Return true

$\textsc{Fin}()$:
$\mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV}}.\textsc{Fin}()$

Adversary $\mathrm{A}_{\mathrm{ds1}}$:

$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\$\; \mathsf{S.C}$; $vk_1 \leftarrow\!\$\; \mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}_1}.\textsc{Init}$; $\mathrm{A}_{\mathrm{snd3}}^{\textsc{Init},\textsc{Pf},\textsc{Vf},\textsc{Fin}}$

$\textsc{Init}$:
Return (crs, vk_1)

$\textsc{Pf}(x)$:
$(sk_2, vk_2) \leftarrow\!\$\; \mathsf{DS}_2.\mathsf{K}$; $\sigma_1 \leftarrow\!\$\; \mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}_1}.\textsc{Sign}((x, vk_2))$
$\mathrm{pf} \leftarrow\!\$\; \Pi.\mathsf{P}(\mathrm{crs}, (x, vk_1, vk_2), \sigma_1)$
$\sigma_2 \leftarrow \mathsf{DS}_2.\mathsf{S}(sk_2, vk_2, (x, \mathrm{pf}))$; Return pf

$\textsc{Vf}(x^*, (\mathrm{pf}^*, vk_2^*, \sigma_2^*))$:
$\sigma_1^* \leftarrow\!\$\; \mathsf{S.X}(\mathrm{crs}, \mathrm{td}, (x^*, vk_1, vk_2^*), \mathrm{pf}^*)$; Return true

$\textsc{Fin}()$:
$\mathbf{G}^{\mathrm{suf}}_{\mathsf{DS}_2}.\textsc{Fin}((x^*, vk_2^*), \sigma_2^*)$

Figure 34: More adversaries for Theorem 10.1.

verified correctly. This means we expect line 16 to find a witness under CV. But, due to line 12, this witness can only be for the second claim in the relation, meaning a signature setting $u_2 = \mathsf{true}$ at line 18. So if $u_2 = \mathsf{false}$, as line 22 says, adversary $\mathrm{A}_{\mathrm{xt1}}$ wins.

Adversary $\mathrm{A}_{\mathrm{ds1}}$ is shown in Figure 34. We have

$$\Pr[\mathrm{G}_3(\mathrm{A}_{\mathrm{snd3}})] \leq \mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}_1}(\mathrm{A}_{\mathrm{ds1}}) \; .$$

This seems quite straightforward to justify due to lines 17, 18 of the games. ▌

The above says that the proof system $\overline{\Pi}$ of Figure 32 satisfies SND3 for $\overline{\mathsf{CV}}$ relative to the simulator of the same Figure. Exercise 10.2 below asks you to show that the same proof system also satisfies XT3 for $\overline{\mathsf{CV}}$. Below, single-query-XT3 means XT3 only for adversaries making just one Ex query.

Exercise 10.2 *Let* Π *be a proof system, and* S *a simulator, that are same-CRS. Let* $\mathsf{DS}_1, \mathsf{DS}_2$ *be signature schemes. Let* $\overline{\mathsf{CV}}$ *be a claim validator. Define claim validator* CV*, proof system* $\overline{\Pi}$

and simulator algorithms $\overline{\mathsf{S}}.\mathsf{C}, \overline{\mathsf{S}}.\mathsf{P}$ *as per Figure 32. You are to extend the simulator* $\overline{\mathsf{S}}$*, specifying algorithm* $\overline{\mathsf{S}}.\mathsf{X}$*, and then prove the following. If* Π *is XT1 for* CV *relative to* S*, signature scheme* DS_1 *is UF and signature scheme* DS_2 *is SUF, then* $\overline{\Pi}$ *is single-query-XT3 for* $\overline{\mathsf{CV}}$ *relative to* $\overline{\mathsf{S}}$*.*

11 Negative results about NIZKs

Particularly with regard to what is coming up next, it is valuable to understand something about the necessity of certain elements of NIZKs. The particular question we ask here is, is the CRS necessary? Can we have a NIZK where the CRS is a constant, like always the empty string?

We will show that if the CRS is generated deterministically (it being the empty string is a special case) then one cannot have both SND1 and ZK. So effectively, interesting NIZKs with a fixed CRS are not possible.

The actual result statement is somewhat more subtle and involved. We said that one cannot have both SND1 and ZK, but clearly this is not always the case; it depends on CV. If the latter always returns false, we can indeed give a proof system for it that satisfies both SND1 and ZK. But trivial claim validators like this are not very interesting. What the result says is that if claim validator CV is "non-trivial," then there is no proof system for it that is both SND1 and ZK.

For this, one needs to define what "trivial" (and this "non-trivial") means. Roughly, a claim validator CV is trivial if, given crs, x, it is easy to test whether or not $x \in \mathsf{TrCl}_{\mathsf{CV}}(\text{crs})$. A proof system for such a claim validator is un-interesting because a verifier can test membership of x in $\mathsf{TrCl}_{\mathsf{CV}}(\text{crs})$ itself, and this provides perfect SND1 and ZK. A proof system is not necessary or helpful when CV is trivial.

In the classical complexity-theoretic setting, "trivial" would correspond to the language being in the complexity class **BPP**. This will not work in our setting. Complexity classes formalize worst-case decision problems. In cryptography, definitions and interest is in instances that can be efficiently found. Thus we will start by formalizing what it means for a claim language TC to have a decision procedure.

Consider an algorithm M that, given crs, x, is trying to decide whether or not $x \in \mathsf{TC}(\text{crs})$, when crs is drawn according to some algorithm CG. We want a measure of how well it does. For this consider game $\mathbf{G}^{\text{dec}}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}$ of Figure 35. The job of the adversary A executing with it is to pick challenge x values for M. But this dec-adversary is restricted. *We ask that it have zero probability of setting* bad *at line 2.* This is a condition, or restriction, on the class of allowed adversaries; ones that do not obey this condition are simply not considered. The restriction says that when A submits a challenge x to its Test oracle, it already knows the answer. However, this answer is not given to M. The latter aims to figure out the answer, and the game returns true if it fails. We let $\mathbf{Adv}^{\text{dec}}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}(\mathrm{A}) = \Pr[\mathbf{G}^{\text{dec}}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}(\mathrm{A})]$.

Theorem 11.1 *Let* CG *be a CRS generator that is* ***deterministic****. Let* Π *be a proof system with* $\Pi.\mathsf{C} = \mathsf{CG}$*. Let* S *be a simulator and* CV *a claim validator. Let* $\mathsf{TC} = \mathsf{TrCl}_{\mathsf{CV}}$*. Let algorithm* M *be as shown in Figure 35. Let A be a dec-adversary. Then we can construct adversaries* $\mathrm{A}_{\text{zk}}, \mathrm{A}_{\text{snd1}}$ *(shown explicitly in Figure 35) such that*

$$\mathbf{Adv}^{\text{dec}}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}(\mathrm{A}) \leq 2 \cdot \mathbf{Adv}^{\text{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}_{\text{zk}}) + \mathbf{Adv}^{\text{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}_{\text{snd1}}) \,. \tag{13}$$

The running times of these adversaries are about the time for the execution of game $\mathbf{G}^{\text{dec}}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}$ *with A.*

Game $\mathbf{G}^{\rm dec}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}$

INIT:

$\mathrm{crs} \leftarrow\!\!\$\ \mathsf{CG}$; Return crs

TEST(x, w):

If ($(x \in \mathsf{TC}(\mathrm{crs}))$ and $(\mathsf{CV}(\mathrm{crs}, x, w) = \mathsf{false})$) then $\mathsf{bad} \leftarrow \mathsf{true}$

$d \leftarrow\!\!\$\ \mathsf{M}(\mathrm{crs}, x)$; Return $\perp$

FIN():

Return ($(x \in \mathsf{TC}(\mathrm{crs})) \neq d$)

Decision algorithm $\mathsf{M}(\mathrm{crs}, x)$:

$(\mathrm{crs}_0, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$

$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}_0, \mathrm{td}, x)$

$d \leftarrow \mathsf{\Pi.V}(\mathrm{crs}_0, x, \mathrm{pf})$

Return d

Adversary $\mathrm{A}_{\rm snd1}$:

$\mathrm{crs} \leftarrow\!\!\$\ \mathbf{G}^{\rm snd1}_{\mathsf{\Pi},\mathsf{TC}}.\mathrm{INIT}$

$(\mathrm{crs}_0, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$

$\mathrm{A}^{\mathrm{INIT},\mathrm{TEST},\mathrm{FIN}}$

INIT:

Return crs

TEST(x, w):

$\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, x)$

$\mathrm{vf} \leftarrow \mathbf{G}^{\rm snd1}_{\mathsf{\Pi},\mathsf{TC}}.\mathrm{VF}(x, \mathrm{pf})$

Return $\perp$

FIN():

$\mathbf{G}^{\rm snd1}_{\mathsf{\Pi},\mathsf{TC}}.\mathrm{FIN}$

Adversary $\mathrm{A}_{\rm zk}$:

$i \leftarrow\!\!\$\ \{1, 2\}$

$\mathrm{A}_{\rm zk,i}^{\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{INIT},\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{PF},\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{FIN}}$

Adversary $\mathrm{A}_{\rm zk,1}$:

$\mathrm{crs} \leftarrow\!\!\$\ \mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{INIT}$

$\mathrm{A}^{\mathrm{INIT},\mathrm{TEST},\mathrm{FIN}}$

INIT:

Return crs

TEST(x, w):

$\mathrm{pf} \leftarrow\!\!\$\ \mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{PF}(x, w)$

$d \leftarrow \mathsf{\Pi.V}(\mathrm{crs}, x, \mathrm{pf})$; Return $\perp$

FIN():

If (d and $\mathsf{CV}(\mathrm{crs}, x, w)$) then $\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{FIN}(1)$

Else $\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{FIN}(0)$

Adversary $\mathrm{A}_{\rm zk,2}$:

$\mathrm{crs}_1 \leftarrow \mathsf{CG}$; $\mathrm{crs} \leftarrow\!\!\$\ \mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{INIT}$

If ($\mathrm{crs} = \mathrm{crs}_1$) then $\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{FIN}(1)$

Else $\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}.\mathrm{FIN}(0)$

Figure 35: Top left: Decidability game. To right: Decision algorithm for Theorem 11.1. Bottom left: Adversaries for theorem. Bottom right: Adversaries called by $\mathrm{A}_{\rm zk}$.

The way to read the Theorem is that, SND1 and ZK both holding means the RHS of Equation (13) is small, so the decision algorithm succeeds, which means CV is trivial.

Proof of Theorem 11.1: Consider the games of Figure 36. We have

$$\mathbf{Adv}^{\rm dec}_{\mathsf{TC},\mathsf{CG},\mathsf{M}}(\mathrm{A}) = \Pr[\mathrm{G}_0(\mathrm{A})] + \Pr[\mathrm{G}_5(\mathrm{A})] .$$

We will bound these in turn.

We have

$$\Pr[\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}[1](\mathrm{A}_{\rm zk,1})] = \Pr[\mathrm{G}_3(\mathrm{A})] = \Pr[\mathrm{G}_4(\mathrm{A})] ,$$

where the second equality is by the (perfect) completeness of $\mathsf{\Pi}$ for CV. Also

$$\Pr[\mathbf{G}^{\rm zk}_{\mathsf{\Pi},\mathsf{CV},\mathsf{S}}[0](\mathrm{A}_{\rm zk,1})] = \Pr[\mathrm{G}_1(\mathrm{A})] = \Pr[\mathrm{G}_4(\mathrm{A})] - \Pr[\mathrm{G}_0(\mathrm{A})] .$$

Games G_0–G_4	Game $\boxed{G_5}$, G_6
INIT:	INIT:
$\text{crs} \leftarrow \mathsf{CG}$; Return crs	$\text{crs} \leftarrow \mathsf{CG}$; $(\text{crs}_0, \text{td}) \leftarrow\$ \mathsf{S.C}$
	If $(\text{crs}_0 \neq \text{crs})$ then
	$\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\text{crs} \leftarrow \text{crs}_0}$
	Return crs
TEST(x, w):	TEST(x, w):
$(\text{crs}_0, \text{td}) \leftarrow\$ \mathsf{S.C}$	$\text{pf} \leftarrow\$ \mathsf{S.P}(\text{crs}, \text{td}, x)$
$\text{pf}_0 \leftarrow\$ \mathsf{S.P}(\text{crs}_0, \text{td}, x)$; $d_0 \leftarrow \Pi.\mathsf{V}(\text{crs}_0, x, \text{pf}_0)$	$d \leftarrow \Pi.\mathsf{V}(\text{crs}, x, \text{pf})$
$\text{pf}_1 \leftarrow\$ \Pi.\mathsf{P}(\text{crs}, x, w)$; $d_1 \leftarrow \Pi.\mathsf{V}(\text{crs}, x, \text{pf}_1)$	Return $\perp$
Return $\perp$	
FIN():	FIN():
Return ($\mathsf{CV}(\text{crs}, x, w)$ and $\neg d_0$) // Game G_0	Return ($\neg\mathsf{CV}(\text{crs}, x, w)$ and d)
Return ($\mathsf{CV}(\text{crs}, x, w)$ and d_0) // Game G_1	
Return ($\mathsf{CV}(\text{crs}, x, w)$ and $\neg d_1$) // Game G_2	
Return ($\mathsf{CV}(\text{crs}, x, w)$ and d_1) // Game G_3	
Return $\mathsf{CV}(\text{crs}, x, w)$ // Game G_4	

Figure 36: Games for proof of Theorem **??**.

So

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(A_{\mathrm{zk},1}) &= \Pr[\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}[1](A_{\mathrm{zk},1})] - \Pr[\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}[0](A_{\mathrm{zk},1})] \\
&= \Pr[G_4(A)] - (\Pr[G_4(A)] - \Pr[G_0(A)]) \\
&= \Pr[G_0(A)] \, .
\end{aligned}$$

Games G_5, G_6 are identical-unti-bad so by Lemma 3.1 we have

$$\begin{aligned}
\Pr[G_5(A)] &= \Pr[G_6(A)] + (\Pr[G_5(A)] - \Pr[G_6(A)]) \\
&\leq \Pr[G_6(A)] + \Pr[G_6(A) \text{ sets } \mathsf{bad}] \, .
\end{aligned}$$

Then we have

$$\begin{aligned}
\Pr[G_6(A)] &\leq \mathbf{Adv}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}(A_{\mathrm{snd1}}) \\
\Pr[G_6(A) \text{ sets } \mathsf{bad}] &\leq \mathbf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(A_{\mathrm{zk},2}) \, .
\end{aligned}$$

Finally, A_{zk} picks $i \leftarrow\$ \{1, 2\}$ and runs $A_{\mathrm{zk,i}}$. ▮

12 Sigma protocols

A Sigma protocol is a 3-move message-exchange between two parties, one called the prover (formalized below as a Sigma prover) and the other called the verifier. It operates as depicted in Figure 37.

A Sigma prover is an algorithm P^* that operates in two stages. *Commitment:* Via $(\text{cmt}, \text{st}) \leftarrow\$ \mathsf{P}^*(\text{crs}, X)$, the first stage takes the common reference string crs and a claim X to return a message cmt (called the commitment) to be sent to the verifier, and state information st to be retained. *Response:* Via $\text{rsp} \leftarrow \mathsf{P}^*(\text{crs}, \text{st}, w, \text{ch})$, the second stage takes crs, state information st from stage

Prover		**Verifier**
Input: crs, X, w		Input: crs, X
$(\mathrm{cmt}, \mathrm{st}) \leftarrow\$ \, \Sigma.\mathsf{P}(\mathrm{crs}, X)$	$\xrightarrow{\mathrm{cmt}}$	
	$\xleftarrow{\mathrm{ch}}$	$c \leftarrow\$ \, \mathsf{ChS}(\mathrm{crs})$
$\mathrm{rsp} \leftarrow \Sigma.\mathsf{P}(\mathrm{crs}, \mathrm{st}, w, \mathrm{ch})$	$\xrightarrow{\mathrm{rsp}}$	$d \leftarrow \Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp}))$

Algorithm $\mathbf{EXEC}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$:

```
(cmt, st) <-$ P*(crs, X)
ch <-$ Σ.ChS(crs)
rsp <- P*(crs, st, w, ch)
Return (cmt, ch, rsp)
```

Figure 37: Operation of a Sigma protocol Σ. The top depicts an interaction between the honest Sigma prover $\Sigma.\mathsf{P}$ and the verifier $\Sigma.\mathsf{V}$. The algorithm at the bottom returns a transcript of an interaction between a (not necessarily honest) Sigma prover P^* and the verifier $\Sigma.\mathsf{V}$.

---

one, a string w (usually, but not always, the witness) and a challenge ch sent by the verifier, to deterministically return a message rsp (called the response) to be sent to the verifier.

A *Sigma protocol* Σ specifies the following algorithms:

- *CRS generation.* Via $\mathrm{crs} \leftarrow\$ \, \Sigma.\mathsf{C}$, the crs-generation algorithm $\Sigma.\mathsf{C}$ (takes no inputs and) returns a common reference string crs.
- *Sigma prover.* $\Sigma.\mathsf{P}$ is a Sigma prover as above.
- *Challenge set.* $\Sigma.\mathsf{ChS}$ associates to crs a finite set $\Sigma.\mathsf{ChS}(\mathrm{crs})$ called the challenge space.
- *Verification.* Via $d \leftarrow \Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp}))$, the deterministic verification algorithm $\Sigma.\mathsf{V}$ produces a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether or not it considers the conversation transcript $(\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp})$ valid.

Algorithm $\mathbf{EXEC}_{\Sigma,\mathsf{P}^*}$ in Figure 37 takes inputs crs, X, w and executes an interaction between Sigma prover P^* and the verifier of Σ. (The Sigma prover P^* could be, but need not be, $\Sigma.\mathsf{P}$, and w could be, but need not be, in $\mathsf{CV}(\mathrm{crs}, X)$.) Given crs, X, w, the algorithm returns a transcript $\mathrm{tr} = (\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp})$ of an interaction between the parties. We say that tr is a $\Sigma(\mathrm{crs}, X)$-accepting transcript if $\Sigma.\mathsf{V}(\mathrm{crs}, X, \mathrm{tr}) = \mathsf{true}$. We let

$$\mathbf{Acc}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w) = \Pr[\Sigma.\mathsf{V}(\mathrm{crs}, X, \mathbf{EXEC}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)) = \mathsf{true}] .$$

This is the probability that $\Sigma.\mathsf{V}(\mathrm{crs}, X, \mathrm{tr}) = \mathsf{true}$ when $\mathrm{tr} \leftarrow\$ \, \mathbf{EXEC}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$. This is the accepting probability of an interaction between P^* and the verifier of Σ on inputs crs, X, w.

We require perfect completeness, although this can be relaxed if necessary. We say that Σ has (perfect) completeness for claim validator CV if $\Sigma.\mathsf{V}(\mathrm{crs}, X, \mathrm{tr}) = \mathsf{true}$ for all transcripts $\mathrm{tr} \in [\mathbf{EXEC}_{\Sigma,\Sigma.\mathsf{P}}(\mathrm{crs}, X, w)]$, all $\mathrm{crs} \in [\Sigma.\mathsf{C}]$, all $X \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$ and all $w \in \mathsf{CV}(\mathrm{crs}, X)$. Put another way, if $\mathsf{CV}(\mathrm{crs}, X, w) = \mathsf{true}$, then $\mathbf{Acc}_{\Sigma,\Sigma.\mathsf{P}}(\mathrm{crs}, X, w) = 1$.

DL Sigma protocol. Our first example is Schnorr's protocol for proving knowledge of a discrete

Prover		**Verifier**
Input: $(p, \mathbb{G}, g), X, w$		Input: $(p, \mathbb{G}, g), X$
$r \leftarrow\$ \mathbb{Z}_p$; $R \leftarrow g^r$	$\xrightarrow{R}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}_p$
$a \leftarrow r + wc$	$\xrightarrow{a}$	
		$d \leftarrow (g^a = RX^c)$

Prover $\Sigma.\mathsf{P}((p, \mathbb{G}, g), X)$:

1 $r \leftarrow\$ \mathbb{Z}_p$; $R \leftarrow g^r$
2 Return (R, r)

Prover $\Sigma.\mathsf{P}((p, \mathbb{G}, g), r, w, c)$:

3 $a \leftarrow r + wc$
4 Return a

Verifier $\Sigma.\mathsf{V}((p, \mathbb{G}, g), X, (R, c, a))$:

5 Return $(g^a = RX^c)$

$\Sigma.\mathsf{ChS}((p, \mathbb{G}, g)) = \mathbb{Z}_p$

Extractor $\Sigma.\mathsf{X}((p, \mathbb{G}, g), X, R, c_1, c_2, a_1, a_2)$:

6 $c \leftarrow (c_1 - c_2)^{-1}$; $w \leftarrow (a_1 - a_2)c$
7 Return w

Simulator $\Sigma.\mathsf{S}((p, \mathbb{G}, g), X)$:

8 $c \leftarrow\$ \mathbb{Z}_p$; $a \leftarrow\$ \mathbb{Z}_p$; $R \leftarrow g^a X^{-c}$
9 Return (R, c, a)

Claim validator $\mathsf{CV}((p, \mathbb{G}, g), X, w)$:

10 Return ($(w \in \mathbb{Z}_p)$ and $(g^w = X)$)

Figure 38: DL Sigma protocol, pictorial (top) and formal (bottom left) descriptions. Bottom right: Sigma extractor, simulator and the claim validator.

---

logarithm [Sch91], shown in Figure 38. The CRS $(p, \mathbb{G}, g)$ consists of a (description of a) group $\mathbb{G}$ of prime order p, and a generator g of $\mathbb{G}$. We regard these as fixed, so that $\Sigma.\mathsf{C}$ is the deterministic algorithm that simply returns this one CRS each time it is invoked. The claim validator is shown in the Figure. In the protocol, the computation of R is in $\mathbb{G}$, while the computation of a is modulo p.

Sigma extractability. Let Σ be a Sigma protocol for claim validator CV. We say that a pair $(\mathrm{cmt}_1, \mathrm{ch}_1, \mathrm{rsp}_1), (\mathrm{cmt}_2, \mathrm{ch}_2, \mathrm{rsp}_2)$ of transcripts is $\Sigma(\mathrm{crs}, X)$-*compatible* if the following are true. (1) Both transcripts are $\Sigma(\mathrm{crs}, X)$-accepting. (2) They have the same commitment, meaning $\mathrm{cmt}_1 = \mathrm{cmt}_2$. (3) They have different challenges, meaning $\mathrm{ch}_1 \neq \mathrm{ch}_2$.

A Sigma-extractor is an algorithm $\Sigma.\mathsf{X}$ with the property that

$$\Sigma.\mathsf{X}(\mathrm{crs}, X, \mathrm{cmt}, \mathrm{ch}_1, \mathrm{ch}_2, \mathrm{rsp}_1, \mathrm{rsp}_2) \in \mathsf{CV}(\mathrm{crs}, X)$$

for all $\mathrm{crs} \in [\Sigma.\mathsf{C}]$, all $X \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$ and all $\mathrm{cmt}, \mathrm{ch}_1, \mathrm{ch}_2, \mathrm{rsp}_1, \mathrm{rsp}_2$ such that the pair of transcripts $(\mathrm{cmt}, \mathrm{ch}_1, \mathrm{rsp}_1), (\mathrm{cmt}, \mathrm{ch}_2, \mathrm{rsp}_2)$ is $\Sigma(\mathrm{crs}, X)$-compatible. That is, suppose $X \in \mathsf{TrCl}_{\mathsf{CV}}$ and suppose $(\mathrm{cmt}_1, \mathrm{ch}_1, \mathrm{rsp}_1)$ and $(\mathrm{cmt}_2, \mathrm{ch}_2, \mathrm{rsp}_2)$ are $\Sigma(\mathrm{crs}, X)$-accepting transcripts. Suppose they have the same commitment, meaning $\mathrm{cmt}_1 = \mathrm{cmt}_2$. Denote this common value by cmt. Suppose they have different challenges, meaning $\mathrm{ch}_1 \neq \mathrm{ch}_2$. Then extractor $\Sigma.\mathsf{X}$, on input $\mathrm{crs}, X, \mathrm{cmt}, \mathrm{ch}_1, \mathrm{ch}_2, \mathrm{rsp}_1, \mathrm{rsp}_2$, returns a valid witness w for X, meaning $\mathsf{CV}(\mathrm{crs}, X, w) = \mathsf{true}$. We say that Σ satisfies Sigma-extractability if it has a Sigma extractor. In the literature, the property is typically called special soundness, but we use a different name since it pertains more to extraction.

Figure 38 shows a Sigma-extractor for the DL protocol. Suppose $(R, c_1, a_1), (R, c_2, a_2)$ is a pair of $\Sigma(\mathrm{crs}, X)$-compatible transcripts. Then $c_1 \neq c_2$, so at line 6 we have $c_1 - c_2 \in \mathbb{Z}_p$ is non-zero. It

Game $\mathbf{G}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}$

$\textsc{Init}()$:
$b \leftarrow\!\!\$\, \{0,1\}$; $\mathrm{crs} \leftarrow\!\!\$\, \Sigma.\mathsf{C}$
Return crs

$\textsc{Pf}(X, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, X, w)$) then return $\perp$
If ($b = 1$) then $\mathrm{tr} \leftarrow\!\!\$\, \mathbf{EXEC}_{\Sigma,\Sigma.\mathsf{P}}(\mathrm{crs}, X, w)$
Else $\mathrm{tr} \leftarrow\!\!\$\, \Sigma.\mathsf{S}(\mathrm{crs}, X)$
Return tr

$\textsc{Fin}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{wi}}_{\Sigma,\mathsf{CV}}$

$\textsc{Init}()$:
$\mathrm{crs} \leftarrow\!\!\$\, \Sigma.\mathsf{C}$; $b \leftarrow\!\!\$\, \{0,1\}$
Return crs

$\textsc{Pf}(X, w_0, w_1)$:
If ($(\mathsf{CV}(\mathrm{crs}, X, w_0) = \mathsf{false})$ or $(\mathsf{CV}(\mathrm{crs}, X, w_1) = \mathsf{false})$)
then return $\perp$
$\mathrm{tr} \leftarrow\!\!\$\, \mathbf{EXEC}_{\Sigma,\Sigma.\mathsf{P}}(\mathrm{crs}, X, w_b)$
Return tr

$\textsc{Fin}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}$

$\textsc{Init}()$:
$\mathrm{crs} \leftarrow\!\!\$\, \Sigma.\mathsf{C}$; Return crs

$\mathrm{CMT}(X, \mathrm{cmt})$:
If ($X \in \mathsf{TC}(\mathrm{crs})$) then return $\perp$
$m \leftarrow m + 1$; $\mathrm{cmt}_m \leftarrow \mathrm{cmt}$; $X_m \leftarrow X$
$\mathrm{ch}_m \leftarrow\!\!\$\, \Sigma.\mathsf{ChS}(\mathrm{crs})$; Return ch_m

$\textsc{Vf}(i, \mathrm{rsp})$:
$\mathrm{vf}_i \leftarrow \Sigma.\mathsf{V}(\mathrm{crs}, X_i, (\mathrm{cmt}_i, \mathrm{ch}_i, \mathrm{rsp}))$
If vf_i then $\mathsf{win} \leftarrow \mathsf{true}$
Return vf_i

$\textsc{Fin}()$:
Return win

Algorithm $\mathbf{REW}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$:
$(\mathrm{cmt}, \mathrm{st}) \leftarrow\!\!\$\, \mathsf{P}^*(\mathrm{crs}, X)$
$\mathrm{ch}_1 \leftarrow\!\!\$\, \Sigma.\mathsf{ChS}(\mathrm{crs})$; $\mathrm{ch}_2 \leftarrow\!\!\$\, \Sigma.\mathsf{ChS}(\mathrm{crs})$
$\mathrm{rsp}_1 \leftarrow\!\!\$\, \mathsf{P}^*(\mathrm{crs}, \mathrm{st}, w, \mathrm{ch}_1)$; $\mathrm{rsp}_2 \leftarrow\!\!\$\, \mathsf{P}^*(\mathrm{crs}, \mathrm{st}, w, \mathrm{ch}_2)$
$\mathrm{vf}_1 \leftarrow \Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}_1, \mathrm{rsp}_1))$
$\mathrm{vf}_2 \leftarrow \Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}_2, \mathrm{rsp}_2))$
Return (vf_1 and vf_2 and $(\mathrm{ch}_1 \neq \mathrm{ch}_2)$)

Figure 39: Top: Games defining zero-knowledge and witness indistinguishability of Sigma Protocol Σ. Bottom: Game defining soundness of Σ, and algorithm for rewinding execution a Sigma Prover P^*.

---

thus has an inverse in the group $\mathbb{Z}_p^*$, which we denote by c. Then w is computed as shown, the operations being in the field $\mathbb{Z}_p$, and returned as the witness.

To see that it works, the assumption that the transcript pair is $\Sigma(\mathrm{crs}, X)$-compatible means that $g^{a_1} = RX^{c_1}$ and $g^{a_2} = RX^{c_2}$, whence $g^{a_1 - a_2} = X^{c_1 - c_2}$ and thus $g^w = X$.

Another way to see it is the following. Let r, x be such that $R = g^r$ and $X = g^x$. Then, the given $g^{a_1} = RX^{c_1}$ and $g^{a_2} = RX^{c_2}$ translate to $a_1 = r + xc_1$ and $a_2 = r + xc_2$. These equations can be solved for the unknown x to give $x = (a_1 - a_2)(c_1 - c_2)^{-1}$, which is what we returned as w.

Zero knowledge. Zero knowledge (ZK) of Σ for CV asks that there be a simulator $\Sigma.\mathsf{S}$ that, given a true statement (meaning, one in $\mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$) can create for it a transcript indistinguishable from one arising from the interaction of the honest prover with the verifier. Unlike for NIZKs, there is no alternative CRS generation algorithm, and no trapdoor for the CRS involved. Formally, consider game $\mathbf{G}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}$ specified in Figure 39. Adversary A can adaptively request transcripts by supplying an instance and a valid witness for it. The transcript is produced either by the interaction between the honest, witness-equipped prover with the verifier, or by the simulator $\Sigma.\mathsf{S}$. The adversary outputs a guess b' as to whether the transcripts were real or simulated. Let $\mathbf{Adv}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}(\mathrm{A})]-1$. The requirement for a Sigma protocol is perfect ZK, meaning $\mathbf{Adv}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}(\mathrm{A}) = 0$ for all A, regardless of their running time. For this definition, we can restrict attention to adversaries making just one query to their PF oracle.

Figure 38 shows the simulator for the DL Sigma protocol Σ. The fact that $\mathbf{Adv}^{\mathrm{zk}}_{\Sigma,\mathsf{CV}}(\mathrm{A}) = 0$ for all A is not formally proved in the literature. Perhaps it is seen as clear. But it is instructive to ask how one might give a rigorous proof. The key element is the following. Fix $w, c \in \mathbb{Z}_p$ and let $X = g^w$. Define $f_{w,c} \colon \mathbb{Z}_p \to \mathbb{G}$ by $f_{w,c}(a) = g^{a-cw} = g^a X^{-c}$. Then $f_{w,c}$ is a bijection.

Witness indistinguishability. This asks that, knowing $X \in \mathsf{TrCl}_{\mathsf{CV}}(\mathrm{crs})$ and knowing two witnesses $w_0, w_1 \in \mathsf{CV}(\mathrm{crs}, x)$, it is hard to tell under which of the two a transcript has been computed. Consider game $\mathbf{G}^{\mathrm{wi}}_{\Sigma,\mathsf{CV}}$ specified in Figure 39. Let $\mathbf{Adv}^{\mathrm{wi}}_{\Sigma,\mathsf{CV}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\mathrm{wi}}_{\Sigma,\mathsf{CV}}(\mathrm{A})] - 1$.

Reset lemma. Algorithm $\mathbf{REW}_{\Sigma,\mathsf{P}^*}$ of Figure 39 takes inputs crs, X, w. It then produces two, related transcripts of the interaction between Sigma Prover P^* and the verifier of Σ on these inputs, returning true if these transcripts are $\Sigma(\mathrm{crs}, X)$-compatible. Let $\mathbf{Rew}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$ be the probability that $\mathbf{REW}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$ returns true. This is the rewinding probability. The Reset Lemma of [BP02] relates the rewinding probability to the acceptance probability. We start with the following special case of Jensen's inequality.

Lemma 12.1 *Let X be a real-valued random variable. Then $\mathbf{E}\left[\mathsf{X}^2\right] \geq \mathbf{E}\left[\mathsf{X}\right]^2$.*

Proof of Lemma 12.1: Let $\mu = \mathbf{E}\left[\mathsf{X}\right]$. The random variable $\mathsf{Y} = (\mathsf{X}-\mu)^2$ is non-negative, and hence its expectation is as well. Using linearity of expectation, we thus have

$$\begin{aligned} 0 \leq \mathbf{E}\left[Y\right] &= \mathbf{E}\left[(\mathsf{X}-\mu)^2\right] = \mathbf{E}\left[\mathsf{X}^2 - 2\mu\cdot\mathsf{X} + \mu^2\right] \\ &= \mathbf{E}\left[\mathsf{X}^2\right] - 2\mu\cdot\mathbf{E}\left[\mathsf{X}\right] + \mu^2 = \mathbf{E}\left[\mathsf{X}^2\right] - 2\mu\cdot\mu + \mu^2 = \mathbf{E}\left[\mathsf{X}^2\right] - \mu^2 \,, \end{aligned}$$

and thus $\mu^2 \leq \mathbf{E}\left[\mathsf{X}^2\right]$. ∎

The following is the Reset Lemma of [BP02]. Its value stems from the rewinding probability $\mathbf{Rew}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$ being also the probability with which the Sigma extractor $\Sigma.\mathsf{X}$ succeeds in finding a witness $w \in \mathsf{CV}(\mathrm{crs}, X)$. This however does not figure explicitly in the statement.

Lemma 12.2 *Let Σ be a Sigma protocol, P^* a Sigma Prover and CV a claim validator. Let $\mathrm{crs} \in [\Sigma.\mathsf{C}]$ and $X, w \in \{0,1\}^*$. Let $\epsilon_{\mathrm{acc}} = \mathbf{Acc}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$ and $\epsilon_{\mathrm{rew}} = \mathbf{Rew}_{\Sigma,\mathsf{P}^*}(\mathrm{crs}, X, w)$. Let $N = |\Sigma.\mathsf{ChS}(\mathrm{crs})|$. Then*

$$\epsilon_{\mathrm{rew}} \geq \epsilon_{\mathrm{acc}} \cdot \left(\epsilon_{\mathrm{acc}} - \frac{1}{N}\right) \tag{14}$$

and

$$\epsilon_{\mathrm{acc}} \leq \frac{1}{N} + \sqrt{\epsilon_{\mathrm{rew}}} \,. \tag{15}$$

Proof of Lemma 12.2: Let

$$\mathsf{X}(\mathrm{cmt}, \mathrm{st}) = \Pr[\Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp})) = \mathsf{true}]$$

where the probability is over ch $\leftarrow\$ \Sigma.\mathsf{ChS}(\mathrm{crs})$ and rsp $\leftarrow\$ \mathsf{P}^*(\mathrm{crs},\mathrm{st},w,\mathrm{ch})$. Regard X as a random variable over the choice of $(\mathrm{cmt},\mathrm{st}) \leftarrow\$ \mathsf{P}^*(\mathrm{crs},X)$. Then

$$\mathbf{E}\left[\mathsf{X}\right] = \epsilon_{\mathrm{acc}} \; .$$

Now we have

$$\begin{aligned}
\epsilon_{\mathrm{rew}} &\geq \mathbf{E}\left[\mathsf{X}\cdot\left(\mathsf{X}-\frac{1}{N}\right)\right] = \mathbf{E}\left[\mathsf{X}^2 - \frac{1}{N}\cdot\mathsf{X}\right] \\
&= \mathbf{E}\left[\mathsf{X}^2\right] - \frac{1}{N}\cdot\mathbf{E}\left[\mathsf{X}\right] & (16)\\
&\geq \mathbf{E}\left[\mathsf{X}\right]^2 - \frac{1}{N}\cdot\mathbf{E}\left[\mathsf{X}\right] & (17)\\
&= \epsilon_{\mathrm{acc}}^2 - \frac{1}{N}\cdot\epsilon_{\mathrm{acc}} = \epsilon_{\mathrm{acc}}\cdot\left(\epsilon_{\mathrm{acc}}-\frac{1}{N}\right) \; .
\end{aligned}$$

Equation (16) is by linearity of expectation. Equation (17) is by Lemma 12.1. This establishes Equation (14).

Letting $\alpha = 1/2N$, from Equation (14) we have

$$\epsilon_{\mathrm{rew}} \geq \epsilon_{\mathrm{acc}}^2 - 2\alpha\epsilon_{\mathrm{acc}} = (\epsilon_{\mathrm{acc}}-\alpha)^2 - \alpha^2$$

and thus

$$\begin{aligned}
\epsilon_{\mathrm{acc}} - \alpha &\leq \sqrt{\alpha^2 + \epsilon_{\mathrm{rew}}} \\
&\leq \alpha + \sqrt{\epsilon_{\mathrm{rew}}} \; ,
\end{aligned}$$

where the last inequality used the fact that $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ for any non-negative real-numbers a,b. This establishes Equation (15). ∎

Soundness. To Sigma protocol Σ and true-claim language TC we associate game $\mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}$ of Figure 39. It is required that a $\mathrm{V}\textsc{f}(i,\mathrm{rsp})$ query of adversary A satisfy $1 \leq i \leq m$. Let $\mathbf{Adv}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}(\mathrm{A}) = \Pr[\mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}(\mathrm{A})]$ be the snd1-advantage of adversary A.

EQDL Sigma protocol. Our next example is a protocol for proving equality of discrete logarithms, shown in Figure 40. The CRS $(p,\mathbb{G})$ consists of a (description of a) group $\mathbb{G}$ of prime order p. We regard these as fixed, so that $\Sigma.\mathsf{C}$ is the deterministic algorithm that simply returns this one CRS each time it is invoked. The claim validator is shown in the Figure.

The following establishes soundness of the EQDL Sigma Protocol Σ. Let P^* be an arbitrary prover and crs $= (p,\mathbb{G})$ a CRS. Let $(g_1, g_2 X_1, X_2)$ be a claim not in the language, meaning $\mathrm{dlog}_{\mathbb{G},g_1}(X_1) \neq \mathrm{dlog}_{\mathbb{G},g_2}(X_2)$. Let w be an arbitrary string. What is the probability that the interaction between P^* (initialized with crs, $(g_1, g_2 X_1, X_2), w$) and the verifier of Σ (initialized with crs, X) is accepting? This is $\mathbf{Acc}_{\Sigma,\mathsf{P}^*}(\mathrm{crs},(g_1,g_2X_1,X_2),w)$. The Proposition says that this quantity is at most $1/p$. Below, q is a bound on the number of queries to CMT plus the number of queries to $\mathrm{V}\textsc{f}$.

Proposition 12.3 *Let Σ be the EQDL Sigma Protocol of Figure 40. Let* A *be an adversary making at most q queries across its* CMT *and* Vf *oracles. Let N be the minimum value of p across all*

Prover		**Verifier**
Input: $(p, \mathbb{G}), (g_1, g_2 X_1, X_2), w$		Input: $(p, \mathbb{G}), (g_1, g_2 X_1, X_2)$
$r \leftarrow_{\$} \mathbb{Z}_p \; ; \; R_1 \leftarrow g_1^r \; ; \; R_2 \leftarrow g_2^r$	$\xrightarrow{(R_1, R_2)}$	
	$\xleftarrow{c}$	$c \leftarrow_{\$} \mathbb{Z}_p$
$a \leftarrow r + wc$	$\xrightarrow{a}$	
		$d \leftarrow (\, (g_1^a = R_1 X_1^c)$ and $(g_2^a = R_2 X_2^c)\,)$

Prover $\Sigma.\mathsf{P}(\mathrm{crs}, (g_1, g_2 X_1, X_2))$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
$r \leftarrow_{\$} \mathbb{Z}_p \; ; \; R_1 \leftarrow g_1^r \; ; \; R_2 \leftarrow g_2^r$
Return $((R_1, R_2), r)$

Prover $\Sigma.\mathsf{P}(\mathrm{crs}, r, w, c)$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
$a \leftarrow r + wc$
Return a

Verifier $\Sigma.\mathsf{V}(\mathrm{crs}, (g_1, g_2 X_1, X_2), ((R_1, R_2), c, a))$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
$d_1 \leftarrow (\, (g_1 \in \mathrm{Gen}(\mathbb{G}))$ and $(g_2 \in \mathrm{Gen}(\mathbb{G}))\,)$
$d_2 \leftarrow (\, (g_1^a = R_1 X_1^c)$ and $(g_2^a = R_2 X_2^c)\,)$
Return (d_1 and d_2)

$\Sigma.\mathsf{ChS}(\mathrm{crs}) = \mathbb{Z}_p$

Extractor $\Sigma.\mathsf{X}(\mathrm{crs}, (g_1, g_2 X_1, X_2), (R_1, R_2), c_1, c_2, a_1, a_2)$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
$c \leftarrow (c_1 - c_2)^{-1} \; ; \; w \leftarrow (a_1 - a_2)c$
Return w

Simulator $\Sigma.\mathsf{S}(\mathrm{crs}, (g_1, g_2 X_1, X_2))$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
$c \leftarrow_{\$} \mathbb{Z}_p \; ; \; a \leftarrow_{\$} \mathbb{Z}_p \; ; \; R_1 \leftarrow g_1^a X_1^{-c} \; ; \; R_2 \leftarrow g_2^a X_2^{-c}$
Return $((R_1, R_2), c, a)$

Claim validator $\mathsf{CV}(\mathrm{crs}, (g_1, g_2 X_1, X_2), w)$:

$(p, \mathbb{G}) \leftarrow \mathrm{crs}$
If $(\, (g_1 \notin \mathrm{Gen}(\mathbb{G}))$ or $(g_2 \notin \mathrm{Gen}(\mathbb{G}))\,)$ then return false
Return $(\, (w \in \mathbb{Z}_p)$ and $(g_1^w = X_1)$ and $(g_2^w = X_2)\,)$

Figure 40: EQDL Sigma protocol, pictorial (top) and formal (bottom left) descriptions. Bottom right: Sigma extractor, simulator and the claim validator.

$\mathrm{crs} = (p, \mathbb{G}) \in [\Sigma.\mathsf{C}]$. *Let* $\mathsf{TC} = \mathsf{TrCl}_{\mathsf{CV}}$ *where* CV *is the claim validator of Figure 40. Then*

$$\mathbf{Adv}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}(\mathrm{A}) \leq \frac{q}{N} \; .$$

Proof of Proposition 12.3: We prove this for $q = 1$. The general case follows by the union bound.

Let $(p, \mathbb{G}) \leftarrow \mathrm{crs}$. Let $(g_1, g_2 X_1, X_2), i, \mathrm{rsp}$ be the VF query. Let $w_1 = \mathrm{dlog}_{\mathbb{G},g_1}(X_1)$ and $w_2 = \mathrm{dlog}_{\mathbb{G},g_2}(X_2)$. Let $(R_1, R_2) = \mathrm{cmt}_i$ be the prior CMT query corresponding to this VF query. Let $r_1 = \mathrm{dlog}_{\mathbb{G},g_1}(R_1)$ and $r_2 = \mathrm{dlog}_{\mathbb{G},g_2}(R_2)$. Let

$$\begin{aligned} S &= \{\, c \in \mathbb{Z}_p \,:\, \exists\, a \in \mathbb{Z}_p \,(\, (g_1^a = R_1 X_1^c) \text{ and } (g_2^a = R_2 X_2^c)\,)\,\} \\ &= \{\, c \in \mathbb{Z}_p \,:\, \exists\, a \in \mathbb{Z}_p \,(\, (a = r_1 + w_1 c) \text{ and } (a = r_2 + w_2 c)\,)\,\} \\ &= \{\, c \in \mathbb{Z}_p \,:\, r_1 + w_1 c = r_2 + w_2 c \,\} \; . \end{aligned}$$

Since $(g_1, g_2 X_1, X_2)$ is assumed to not be in $\mathsf{TC}(\mathrm{crs})$, we have $w_1 \neq w_2$. So $x = w_1 - w_2 \neq 0$, and thus has an inverse $y \in \mathbb{Z}_p^*$. Let $c^* = (r_2 - r_1)y$. Then S is the singleton set consisting of c^*. The verifier accepts only if $\mathrm{ch}_i \in S$, which means if $\mathrm{ch}_i = c^*$, which happens with probability $1/p$ because ch_i is chosen at random from $\mathbb{Z}_p$ by CMT. ▮

13 NIZKs in the Random Oracle Model

Applying the Fiat-Shamir transform to Sigma protocols results in ROM (Random Oracle Model) NIZKs. We start with definitions about these following [BR93].

ROM Proof systems. A ROM proof system Π specifies, as before, a CRS generation algorithm $\Pi.\mathsf{C}$. It also associates to each crs $\in [\Pi.\mathsf{C}]$ a set $\Pi.\mathsf{H}(\text{crs})$ of functions. Now $\Pi.\mathsf{P}(\text{crs}, \cdot, \cdot)$ and $\Pi.\mathsf{V}(\text{crs}, \cdot, \cdot)$ get oracle access to a function $\mathrm{H} \in \Pi.\mathsf{H}(\text{crs})$. Games specifying security attributes will draw H at random from $\Pi.\mathsf{H}(\text{crs})$, so that H is the random oracle.

Completeness is required for all choices of H. Thus we say that Π has (perfect) completeness for CV if $\Pi.\mathsf{V}^{\mathrm{H}}(\text{crs}, x, \Pi.\mathsf{P}^{\mathrm{H}}(\text{crs}, x, w)) = \mathsf{true}$ for all crs $\in [\Pi.\mathsf{C}]$, all $x \in \mathsf{TrCl}_{\mathsf{CV}}(\text{crs})$ and all $w \in \mathsf{CV}(\text{crs}, x)$ and all $\mathrm{H} \in [\Pi.\mathsf{H}(\text{crs})]$.

ZK and WI. Game $\mathbf{G}^{\text{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}$ is shown in Figure 41. Table RT is a global variable in the game that starts out everywhere $\bot$ and at any point contains the current content of the random oracle. The simulator-prover $\mathsf{S.P}$ gets RT as input, and can modify its contents, which then get reflected in further usage of the table across the game. This modification is the step of "programming" the random oracle. Let $\mathbf{Adv}^{\text{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\text{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})] - 1$.

WI is as usual simpler since no simulator is involved. Game $\mathbf{G}^{\text{wi}}_{\Pi,\mathsf{CV}}$ is shown in Figure 41 and we let $\mathbf{Adv}^{\text{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A}) = 2\Pr[\mathbf{G}^{\text{wi}}_{\Pi,\mathsf{CV}}(\mathrm{A})] - 1$.

SND1 and XT1. Game $\mathbf{G}^{\text{snd1}}_{\Pi,\mathsf{TC}}$ is shown in Figure 41. Let $\mathbf{Adv}^{\text{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A}) = \Pr[\mathbf{G}^{\text{snd1}}_{\Pi,\mathsf{TC}}(\mathrm{A})]$.

Extractability is more troublesome, and it is not clear what definition strikes the best balance between the properties desired and what can be obtained via Lemma 12.2. I am not sure what is a good definition. To have one to consider, I have given game $\mathbf{G}^{\text{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}$ of Figure 41. It is required that the adversary playing this game is deterministic. Its coins are represented by ρ, chosen by the game and returned to the adversary. This is done so that ρ can also be provided to the extractor $\mathsf{S.X}$. Let $\mathbf{Adv}^{\text{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A}) = \Pr[\mathbf{G}^{\text{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}(\mathrm{A})]$.

NIZKs from Sigma protocols. Let Σ be a Sigma protocol. We associate to it a ROM proof system $\Pi\Sigma$ called a Sigma NIZK. Its CRS generator, prover and verifier algorithms are shown in Figure 42. The set of hash functions $\Pi\Sigma.\mathsf{H}(\text{crs})$ it associates to crs is the set of all functions $\mathrm{H}\colon \{0,1\}^* \to \Sigma.\mathsf{ChS}(\text{crs})$ whose range is the challenge space associated to crs, so that a random oracle returns random challenges.

SND1 of Sigma NIZK. The following says that the Sigma NIZK inherits the soundness of the Sigma protocol. However the number of oracle queries of the constructed snd1 adversary depends on the number of random-oracle queries of the starting one, and, when one combines the following with Proposition 12.3, this results in a bound that is linear in the number of random-oracle queries. Thus, while Equation (18) looks tight, there is actually a significant degradation in concrete security in making the protocol non-interactive. This does not always seem recognized [DGS+18].

Proposition 13.1 *Let Σ be a Sigma protocol, and $\Pi\Sigma$ the associated Sigma NIZK as per Figure 42. Let TC be a true-claim language. Let $\mathrm{A}_{\Pi\Sigma}$ be an adversary making Q_{VF} queries to its* VF *oracle and Q_{RO} queries to its* RO *oracle, and let $q = Q_{\mathrm{VF}} + Q_{\mathrm{RO}}$. Then we can construct an adversary*

Game $\mathbf{G}^{\mathrm{zk}}_{\Pi,\mathsf{CV},\mathsf{S}}$

$\textsc{Init}()$:
$b \leftarrow\!\!\$\ \{0,1\}$; $\mathrm{H} \leftarrow\!\!\$\ \Pi.\mathsf{H}(\mathrm{crs})$
If $(b = 1)$ then $\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$
Else $(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$
Return crs

$\textsc{Pf}(X, w)$:
If (not $\mathsf{CV}(\mathrm{crs}, X, w)$) then return $\perp$
If $(b = 1)$ then $\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}^{\mathrm{RO}}(\mathrm{crs}, X, w)$
Else $\mathrm{pf} \leftarrow\!\!\$\ \mathsf{S.P}(\mathrm{crs}, \mathrm{td}, \mathrm{RT}, X)$
Return pf

$\mathrm{RO}(c)$:
If $(\mathrm{RT}[c] = \perp)$ then $\mathrm{RT}[c] \leftarrow \mathrm{H}(c)$
Return $\mathrm{RT}[c]$

$\textsc{Fin}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{wi}}_{\Pi,\mathsf{CV}}$

$\textsc{Init}()$:
$\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$; $b \leftarrow\!\!\$\ \{0,1\}$
Return crs

$\textsc{Pf}(X, w_0, w_1)$:
If ($(\mathsf{CV}(\mathrm{crs}, X, w_0) = \mathsf{false})$ or $(\mathsf{CV}(\mathrm{crs}, X, w_1) = \mathsf{false})$)
  then return $\perp$
$\mathrm{pf} \leftarrow\!\!\$\ \Pi.\mathsf{P}^{\mathrm{RO}}(\mathrm{crs}, X, w_b)$
Return pf

$\mathrm{RO}(c)$:
$y \leftarrow \mathrm{H}(c)$; Return y

$\textsc{Fin}(b')$:
Return $(b' = b)$

Game $\mathbf{G}^{\mathrm{snd1}}_{\Pi,\mathsf{TC}}$

$\textsc{Init}()$:
$\mathrm{crs} \leftarrow\!\!\$\ \Pi.\mathsf{C}$; Return crs

$\textsc{Vf}(X, \mathrm{pf})$:
If $(X \in \mathsf{TC}(\mathrm{crs}))$ then return false
$\mathrm{vf} \leftarrow \Pi.\mathsf{V}^{\mathrm{RO}}(\mathrm{crs}, X, \mathrm{pf})$
If vf then $\mathsf{win} \leftarrow \mathsf{true}$
Return vf

$\mathrm{RO}(c)$:
$y \leftarrow \mathrm{H}(c)$; Return y

$\textsc{Fin}()$:
Return win

Game $\mathbf{G}^{\mathrm{xt1}}_{\Pi,\mathsf{CV},\mathsf{S}}$

$\textsc{Init}()$:
$(\mathrm{crs}, \mathrm{td}) \leftarrow\!\!\$\ \mathsf{S.C}$; $\rho \leftarrow\!\!\$\ \{0,1\}^{\mathsf{A.rl}}$; $st \leftarrow \varepsilon$
Return crs, ρ

$\textsc{Ex}(X, \mathrm{pf})$:
If $(\Pi.\mathsf{V}(\mathrm{crs}, X, \mathrm{pf}) = \mathsf{false})$ then return false
$(w, st) \leftarrow\!\!\$\ \mathsf{S.X}^{\mathrm{RO}}(\mathrm{crs}, \mathrm{td}, st, \rho, X, \mathrm{pf})$
$\mathrm{vf} \leftarrow \mathsf{CV}(\mathrm{crs}, X, w)$
If (not vf) then $\mathsf{win} \leftarrow \mathsf{true}$
Return vf

$\mathrm{RO}(c)$:
$y \leftarrow \mathrm{H}(c)$; Return y

$\textsc{Fin}()$:
Return win

Figure 41: Top: Games defining zero-knowledge (relative to simulator S) and witness indistinguishability of ROM proof system Π. Bottom: Games defining soundness and extractability of ROM proof system Π.

A_{Σ} *such that*

$$\mathbf{Adv}^{\mathrm{snd1}}_{\Pi\Sigma,\mathsf{TC}}(\mathrm{A}_{\Pi\Sigma}) \leq \mathbf{Adv}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}(\mathrm{A}_{\Sigma}) \ . \tag{18}$$

Adversary A_{Σ} *makes at most* q *queries to its* CMT *oracle and at most* q *queries to its* $\textsc{Vf}$ *oracle. Its running time is about that of* $\mathrm{A}_{\Pi\Sigma}$.

Proof of Proposition 13.1: The adversary is shown in Figure 43. ∎

CRS generator $\Pi\Sigma.\mathsf{C}$:

$\mathrm{crs} \leftarrow\!\!\$\ \Sigma.\mathsf{C}$; Return crs

Prover $\Pi\Sigma.\mathsf{P}^{\mathrm{H}}(\mathrm{crs}, X, w)$:

$(\mathrm{cmt}, \mathrm{st}) \leftarrow\!\!\$\ \Sigma.\mathsf{P}(\mathrm{crs}, X)$

$\mathrm{ch} \leftarrow \mathrm{H}((X, \mathrm{cmt}))$

$\mathrm{rsp} \leftarrow \Sigma.\mathsf{P}(\mathrm{crs}, \mathrm{st}, w, \mathrm{ch})$

Return $(\mathrm{cmt}, \mathrm{rsp})$

Verifier $\Pi\Sigma.\mathsf{V}^{\mathrm{H}}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{rsp}))$:

$\mathrm{ch} \leftarrow \mathrm{H}((X, \mathrm{cmt}))$

Return $\Sigma.\mathsf{V}(\mathrm{crs}, X, (\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp}))$

CRS simulator $\mathsf{S.C}$:

$\mathrm{crs} \leftarrow\!\!\$\ \Sigma.\mathsf{C}$

$\mathrm{td} \leftarrow \varepsilon$

Return $(\mathrm{crs}, \mathrm{td})$

Proof simulator $\mathsf{S.P}(\mathrm{crs}, \mathrm{td}, \mathrm{RT}, X)$:

$(\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp}) \leftarrow\!\!\$\ \Sigma.\mathsf{S}(\mathrm{crs}, X)$

$\mathrm{RT}[(X, \mathrm{cmt})] \leftarrow \mathrm{ch}$

Return $(\mathrm{cmt}, \mathrm{rsp})$

Figure 42: Sigma NIZK. Left: CRS generator, prover and verifier. Right: Simulator.

Adversary A_{Σ}:

$\mathrm{crs} \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}.\textsc{Init}$

$\mathrm{A}_{\Pi\Sigma}^{\textsc{Init},\textsc{Vf},\mathrm{RO},\textsc{Fin}}$

$\textsc{Init}$:

Return crs

$\textsc{Vf}(X, (\mathrm{cmt}, \mathrm{rsp}))$:

$\mathrm{ch} \leftarrow\!\!\$\ \mathrm{RO}((\mathrm{crs}, X, \mathrm{cmt}))$

If $(X, \mathrm{cmt}) \in B$ then return false

$\mathrm{vf} \leftarrow \mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}.\textsc{Vf}(\mathrm{Id}[X, \mathrm{cmt}], \mathrm{rsp})$

$\mathrm{RO}((X, \mathrm{cmt}))$:

If $(\mathrm{RT}[(X, \mathrm{cmt})] \neq \perp)$ then return $\mathrm{RT}[(X, \mathrm{cmt})]$

$\mathrm{ch} \leftarrow\!\!\$\ \mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}.\mathrm{CMT}(X, \mathrm{cmt})$

If $(\mathrm{ch} \neq \perp)$ then $m \leftarrow m + 1$; $\mathrm{Id}[X, \mathrm{cmt}] \leftarrow m$

Else $B \leftarrow B \cup \{(X, \mathrm{cmt})\}$; $\mathrm{ch} \leftarrow\!\!\$\ \Sigma.\mathsf{ChS}(\mathrm{crs})$

$\mathrm{RT}[(X, \mathrm{cmt})] \leftarrow \mathrm{ch}$; Return $\mathrm{RT}[(X, \mathrm{cmt})]$

$\textsc{Fin}()$:

$\mathbf{G}^{\mathrm{snd1}}_{\Sigma,\mathsf{TC}}.\textsc{Fin}$

Figure 43: Adversary for Proposition 13.1.

ZK of Sigma NIZK. The simulator S for the Sigma NIZK is shown in Figure 42. The crs is the real one, and the there is no trapdoor. (Formally it is the empty string.) The proof simulator runs the Sigma simulator to get a transcript $(\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp})$. It then programs the random oracle by setting the value of table RT at entry $(\mathrm{crs}, X, \mathrm{cmt})$ to be ch. Recall that RT is a game global variable, so this assignment is now reflected in further uses of the table in the game. In case the table already had a value at $(\mathrm{crs}, X, \mathrm{cmt})$, the simulator just redefines it. This is an error condition that will show up in the analysis below.

For the analysis, we define the commitment collision probability $\Sigma.\mathsf{CC}$ of a Sigma protocol Σ. This

Games G_0, $\boxed{\mathrm{G}_1}$

$\textsc{Init}()$:

$b \leftarrow\$ \{0,1\}$; $\mathrm{crs} \leftarrow\$ \Sigma.\mathsf{C}$

Return crs

$\textsc{Pf}(X, w)$:

If (not $\mathsf{CV}(\mathrm{crs}, X, w)$) then return $\perp$

$(\mathrm{cmt}, \mathrm{ch}, \mathrm{rsp}) \leftarrow\$ \Sigma.\mathsf{S}(\mathrm{crs}, X)$

If $(\mathrm{RT}[(X, \mathrm{cmt})] \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{ch} \leftarrow \mathrm{RT}[(X, \mathrm{cmt})] \text{ ; } \mathrm{rsp} \leftarrow\$ \mathrm{RSP}(\mathrm{crs}, X, w, \mathrm{cmt}, \mathrm{ch})}$

$\mathrm{RT}[(X, \mathrm{cmt})] \leftarrow \mathrm{ch}$

Return $(\mathrm{cmt}, \mathrm{rsp})$

$\mathrm{RO}((X, \mathrm{cmt}))$:

If $(\mathrm{RT}[(X, \mathrm{cmt})] = \perp)$ then $\mathrm{RT}[(X, \mathrm{cmt})] \leftarrow\$ \Sigma.\mathsf{ChS}(\mathrm{crs})$

Return $\mathrm{RT}[(X, \mathrm{cmt})]$

$\textsc{Fin}(b')$:

Return $(b' = 1)$

Figure 44: Games for proof of Proposition 13.2.

---

is done in two steps. First, let $\Sigma.\mathsf{CC}(\mathrm{crs}, X)$ be the probability that $\mathrm{cmt}_1 = \mathrm{cmt}_2$ when we execute

$$(\mathrm{cmt}_1, \mathrm{st}_1) \leftarrow\$ \Sigma.\mathsf{P}(\mathrm{crs}, X) \text{ ; } (\mathrm{cmt}_2, \mathrm{st}_2) \leftarrow\$ \Sigma.\mathsf{P}(\mathrm{crs}, X) \ ,$$

the two executions being independent. Then, let $\Sigma.\mathsf{CC}(\mathrm{crs})$ be the maximum, over all X, of $\Sigma.\mathsf{CC}(\mathrm{crs}, X)$. As examples, $\Sigma.\mathsf{CC}(\mathrm{crs})$ is $1/p$ for both the DL and the EQDL Sigma protocols, where p is the prime in crs. Finally let $\Sigma.\mathsf{CC}$ be the expected value of $\Sigma.\mathsf{CC}(\mathrm{crs})$, taken over $\mathrm{crs} \leftarrow\$ \Sigma.\mathsf{C}$.

Proposition 13.2 *Let Σ be a Sigma protocol and $\Pi\Sigma$ the associated Sigma NIZK as in Figure 42. Let CV be a claim validator. Let simulator S be as in Figure 42. Let A be an adversary making Q_{PF} queries to its* $\textsc{Pf}$ *oracle and Q_{RO} queries to its* RO *oracle. Then*

$$\mathbf{Adv}^{\mathrm{zk}}_{\Pi\Sigma,\mathsf{CV},\mathsf{S}}(\mathrm{A}) \leq Q_{\mathrm{PF}} \cdot (Q_{\mathrm{RO}} + Q_{\mathrm{PF}} - 1) \cdot \Sigma.\mathsf{CC} \ .$$

Proof of Proposition 13.2: Let ω denote the coins of the first stage of $\Sigma.\mathsf{P}$. Let $C(\mathrm{crs}, X, \mathrm{cmt})$ be the set of all ω such the first component of the pair $\Sigma.\mathsf{P}(\mathrm{crs}, X; \omega)$ is cmt. Then let RSP be the following algorithm:

Algorithm $\mathrm{RSP}(\mathrm{crs}, X, w, \mathrm{cmt}, \mathrm{ch})$

$\omega \leftarrow\$ C(\mathrm{crs}, X, \mathrm{cmt})$

$(\mathrm{cmt}, \mathrm{st}) \leftarrow \Sigma.\mathsf{P}(\mathrm{crs}, X; \omega)$

$\mathrm{rsp} \leftarrow\$ \Sigma.\mathsf{P}(\mathrm{crs}, \mathrm{st}, w, \mathrm{ch})$

Return rsp

Consider the games of Figure 44. Then

$$\begin{aligned}\mathbf{Adv}^{\mathsf{zk}}_{\Pi\Sigma,\mathsf{CV},\mathsf{S}}(\mathrm{A}) &= \Pr[\mathrm{G}_1(\mathrm{A})] - \Pr[\mathrm{G}_0(\mathrm{A})] \\ &\leq \Pr[\mathrm{G}_0(\mathrm{A}) \text{ sets } \mathsf{bad}] \\ &\leq Q_{\mathrm{PF}} \cdot (Q_{\mathrm{RO}} + Q_{\mathrm{PF}} - 1) \cdot \Sigma.\mathsf{CC}\ .\end{aligned}$$

This omits a lot of details that need to be filled in. ▌

14 Acknowledgments

Thanks to Lad Steffko, Xiaohan Fu, Zaki Siddiqui, Zhuowen Zou (students, CSE 208 Wi20) for comments and corrections.

References

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010. (Cited on page 4.)

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type II pairings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 390–407. Springer, Heidelberg, August 2014. (Cited on page 4.)

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004. (Cited on page 30.)

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014. (Cited on page 4.)

[BDSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991. (Cited on page 4.)

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. (Cited on page 29.)

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. (Cited on page 4.)

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990. (Cited on page 4, 42.)

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993. (Cited on page 19.)

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005. (Cited on page 29, 31.)

[BH15] Mihir Bellare and Viet Tung Hoang. Adaptive witness encryption and asymmetric password-based cryptography. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 308–331. Springer, Heidelberg, March / April 2015. (Cited on page 9.)

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. (Cited on page 25.)

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. (Cited on page 12.)

[BL13] Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2013. (Cited on page 9.)

[BMT14] Mihir Bellare, Sarah Meiklejohn, and Susan Thomson. Key-versatile signatures and applications: RKA, KDM and joint enc/sig. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 496–513. Springer, Heidelberg, May 2014. (Cited on page 40.)

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, August 2002. (Cited on page 62.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Cited on page 65.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. (Cited on page 10, 11.)

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, June 1996. (Cited on page 4.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Cited on page 4.)

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018. (Cited on page 65.)

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, December 2010. (Cited on page 40.)

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, October 1992. (Cited on page 19.)

[EG14] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014. (Cited on page 4.)

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999. (Cited on page 4.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 39.)

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. (Cited on page 4, 19.)

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. (Cited on page 8.)

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006. (Cited on page 25, 32, 33, 34, 36, 37.)

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006. (Cited on page 52.)

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. (Cited on page 4.)

[Gro15] Jens Groth. Efficient fully structure-preserving signatures for large messages. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 239–259. Springer, Heidelberg, November / December 2015. (Cited on page 4.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. (Cited on page 4.)

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, January 1998. (Cited on page 4.)

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992. (Cited on page 31.)

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. (Cited on page 19.)

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. (Cited on page 60.)