2021 IEEE 34th Computer Security Foundations Symposium (CSF) | 978-1-7281-7607-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSF51468.2021.00014

Machine-Checking Unforgeability Proofs for Signature Schemes with Tight Reductions to the Computational Diffie-Hellman Problem

François Dupressoir
University of Bristol

Sara Zain
University of Surrey

***Abstract*—Digital signatures based on the Discrete Logarithm (DL) problem often suffer from long signature sizes, and reductions made loose by the use of Pointcheval and Stern's Forking Lemma. At EUROCRYPT 2003, Goh and Jarecki provided the first forking-free proof of unforgeability for a DL-based signature scheme—rooting its security in the hardness of the Computational Diffie-Hellman problem in the random oracle model. In this paper, we present and discuss the first machine-checked proofs for DL-based signature schemes reducing tightly to CDH, produced using EasyCrypt. We craft our proofs around a *shim* which reduces the local proof effort, and helps us identify patterns that can be easily adapted to similar tightly-secure DL-based schemes.**

I. Introduction

The EDL signature scheme—based on the Discrete Logarithm (DL) problem—was independently proposed without proof by Chaum and Pedersen [1] and Jakobsson and Schnorr [2]. Goh and Jarecki [3] propose a proof in the random oracle model that EDL is existentially unforgeable under chosen-message attacks when constructed over a group in which the Computational Diffie-Hellman (CDH) problem is hard. Crucially, Goh and Jarecki's proposed security proof does not make use of Pointcheval and Stern's *Forking Lemma* [4]: although EDL signatures contain a Schnorr proof, it is used as a zero-knowledge proof of discrete logarithm equality, rather than as a proof of knowledge. This allows a much tighter reduction than most other DL-based signature schemes, theoretically supporting shorter signatures.

Chevallier-Mames [5] presents an improvement on EDL which reduces the size of signatures and allows cost-free use of *coupons*. The scheme (denoted CM in the following) relies on similar techniques to avoid relying on the forking lemma, but interestingly still relies on the special soundness property of Schnorr proofs.

A. Our Contributions

Our main contribution is the first machine-checked proof for a signature scheme with a tight reduction to CDH. We formalize proofs for both the EDL (Section III) and CM (Section IV) signatures schemes, and make them publicly available.[1] We reformulate Goh and Jarecki [3] and Chevallier-Mames's [5] pen and paper proofs, and mechanise them in EasyCrypt. Like the original results by Goh and Jarecki [3] and Chevallier-Mames [5], our theorems are concrete—rather than asymptotic—security statements. However, the original proofs were direct reductions; ours follows the methodology based on sequences of games advocated by Shoup [6].

This, and the formalization effort, allow us to identify proof principles that we believe are more general, and could be applied more broadly. In developing the proof of security for EDL, we develop a *shim*, used in all intermediate steps of the proof, and whose main interest is in reducing the amount of boilerplate proof. We reuse *the same shim* with very minor tweaks for the CM scheme, whose proof differs only in probability bounding and reduction steps. Section V closes with discussions of further potential generalizations as well as or related and future work.

Due to very minor mistakes and omissions in the original proof by Goh and Jarecki [3], we cannot prove that the bound given originally holds; we prove a very close bound instead, similar to that given for EDL by Chevallier-Mames [5], with minor amendments to address formal details (discussed where relevant). Our formal theorem for the CM scheme is in line with that given originally.

II. Background and Overview

In this section, we review certain background definitions of relevance to this paper's discussions.

A. Code-Based Game-Playing Proofs and Exact Security

Cryptographic proofs often rely on complexity theoretic *reductions*, constructing an adversary against some hardness (or statistical) *assumption* from an adversary against the construction under study.

If early cryptographic proofs were indeed presented much like complexity theoretic reductions, the complexity of modern assumptions and constructions, the difficulty of reasoning rigorously about probabilistic algorithms, and the desire to use security proofs to inform parameter selection (and in particular key sizes) have led to a shift towards code-based game-playing security proofs [6]. These proofs still construct a reduction, but do so step-by-step—with each step called a *game*, allowing careful reasoning about the relation between the probabilities of events in successive games, and isolating complex events whose probability must be reasoned about.

[1] https://gitlab.org/ec-zksigs/edl.git

Further, assumptions, constructions, security notions, and intermediate games are expressed as simple programs. This provides clarity in the definitions, but also helps reason about the *complexity* of the reduction and its *tightness*, stepping away from asymptotic results and providing concrete (or exact) security claims for given security parameters.

In this paper, we do not use the full power of the methodology. Instead, we will only rely on proving statements of the following form, with $\mathcal{D}_1$ and $\mathcal{D}_2$ distributions, and E_1 and E_2 events defined over the relevant distribution's support.

$$\Pr[\mathcal{D}_1 : E_1] \leq \Pr[\mathcal{D}_2 : E_2]$$

In places, we will also prove statements that involve some additional *failure event* F_2 defined over the support of $\mathcal{D}_2$.

$$\Pr[\mathcal{D}_1 : E_1] \leq \Pr[\mathcal{D}_2 : E_2] + \Pr[\mathcal{D}_2 : F_2]$$

We will do so simply by proving the following and applying the union bound.

$$\Pr[\mathcal{D}_1 : E_1] \leq \Pr[\mathcal{D}_2 : E_2 \vee F_2]$$

The distributions we consider are quite often defined by probabilistic programs with specified inputs. Code-based game-playing proofs, and their formalization in EasyCrypt, leverage this by allowing a proof relating two distributions to be lifted from a proof relating the two programs that produce them.

B. EasyCrypt

EasyCrypt[2] [7] is an interactive proof assistant designed to construct machine-checked game-based cryptographic proofs. Cryptographic games and definitions are modelled as probabilistic procedures (with random sampling, conditional statements, loops and procedure calls) over a language of expressions that can be user-extended with various mathematical and data structures.

In addition to the usual features of an interactive proof assistant, EasyCrypt features specialized program logics to reason about probabilistic equivalences between programs (in the *probabilistic Relational Hoare Logic*; pRHL), and to directly reason about probabilities of events in given programs (in the *probabilistic Hoare Logic*; or pHL). We now explain how statements in these logics map to the practice of game-based proofs.

1) pRHL Judgments: A pRHL judgment of the form

$$\{\Phi\}\ c_1 \sim c_2\ \{\Psi\}$$

where c_1 and c_2 are programs, and the *precondition* and *postcondition* Φ and Ψ (respectively) are *relations* between the memories of c_1 and c_2, can be used to prove—for any pair of memories m_1, m_2 such that $m_1\ \Phi\ m_2$—that

$$\Pr[c_1 \,@\, m_1 : E_1] \leq \Pr[c_2 \,@\, m_2 : E_2]$$

whenever for any pair of memories m'_1 and m'_2, such that $m'_1\ \Psi\ m'_2$, we have $E_1(m'_1) \Rightarrow E_2(m'_2)$. The notation $c \,@\, m$ defines a distribution over final states by defining some code c and an initial memory m in which that code is executed. In the following and in our formal definitions, we always consider top-level games whose behaviour is independent of the initial memory (by ensuring that all variables are defined before use). We therefore omit initial memories where possible in the remainder of this paper.

pRHL for equivalence up to failure: For readers who wish to match paper statements here to the formal development, we note that a single pRHL judgement can be used to discharge several statements on probabilities. Anticipating slightly on discussions of the proofs, this is useful in proving statements of the following form when the failure event F is defined over the support of both distributions.

$$\Pr[c_1 \,@\, m_1 : E_1] \leq \Pr[c_2 \,@\, m_2 : E_2] + \Pr[c_1 \,@\, m_1 : F]$$

Indeed, to obtain such a result, a single pRHL judgement—such that the post-condition Ψ implies both that the failure event F occurs with the same probability in c_1 and c_2, and that whenever E_1 holds in c_1, then either E_2 or F occurred in c_2—is sufficient.[3]

2) pHL Judgments: A pHL judgment of the form

$$\{\Phi\}\ c\ \{\Psi\} \diamond b$$

where c is a program, the precondition and postcondition Φ and Ψ are predicates over the memory of c, $b \in [0,1]$ is some probability bound, and $\diamond \in \{=, \leq, \geq\}$ is some relation, can be used to prove—for any memory m such that $\Phi\ m$—that

$$\Pr[c \,@\, m : \Psi] \diamond b$$

Proofs of such statements are relatively straightforward when the event Ψ occurs in the program's main component, but involve invocations of the *failure event lemma* when the event is triggered by oracle queries. The failure event lemma, in the proof we describe below, allows us to lift a bound on the probability of an event occurring during any one oracle query into a bound on the probability of that event occurring during the run of the experiment.

As discussed in relation to game-based proofs, these two ingredients suffice to follow the proofs given below. We note that, although we do not give details, all statements below are given proofs that are fully machine-checked in EasyCrypt. We instead focus our writing on giving an intuition of the reasoning formalised in those machine-checked proofs, and of the aspects of it that might be abstracted into higher-level proof principles.

C. Mathematical Preliminaries

To better support—and in fact add value to—the formalization, we carry out our formal proof on abstract mathematical objects, without specifying their implementation. As shown

[2] https://www.easycrypt.info

[3] The full formal interpretation of pRHL judgments—as originally given by Barthe et al. [8]—allows other deductions, which are not used in this paper.

```
$\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}}(G, g_G, n)$
$\mathsf{a} \leftarrow\!\!\$\; \mathbb{F}_n$
$\mathsf{b} \leftarrow\!\!\$\; \mathbb{F}_n$
$\mathsf{r} \leftarrow \mathcal{A}_{G,g_G,n}(g_G^{\mathsf{a}}, g_G^{\mathsf{b}})$
```

Fig. 1. The CDH experiment.

```
$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{S},\mathcal{F}}()$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{S}.\mathsf{KGen}()$
$(\tilde{\mathsf{m}}, \tilde{\sigma}) \leftarrow \mathcal{F}^{\mathcal{S}.\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$
$\mathsf{b} \leftarrow \mathcal{S}.\mathsf{Ver}_{\mathsf{pk}}(\tilde{\mathsf{m}}, \tilde{\sigma})$
```

Fig. 2. The EUF-CMA experiment.

```
$\mathcal{H}_d(x)$
if $x \notin \mathsf{H}$
    $\mathsf{H}[x] \leftarrow\!\!\$\; d$
return $\mathsf{H}[x]$
```

Fig. 3. A Random Oracle that lazily samples its responses from some distribution d. H is a map or association list, initially empty.

by Almeida, Barbosa, Barthe and Dupressoir [9], [10], this does not preclude further refinements, even all the way down to considering implementation adversaries. More importantly, the proof formalized then applies to all valid instantiations of the abstract objects, provided the refinement ensures that the expected interface is respected.[4]

We consider constructions that rely on a cyclic group $\mathbb{G}$ of prime order q, and on a specific generator g of $\mathbb{G}$ agreed upon ahead of time. $\mathbb{G}$, q and g are assumed to be public, and known—in particular—to the adversary. We use multiplicative notation for the group $\mathbb{G}$, denoting with $1_{\mathbb{G}}$ (or simply 1 when unambiguous) its identity element, and with $\times$ its operation. We use exponents to denote iterations of $\times$ but take exponents directly in the field $\mathbb{F}_q$ such that field addition $(+)$ and multiplication $(\cdot)$ in the exponent are *implicitly* carried out modulo q. We use $\cdot^{-1}$ for field inversion. Multiplication symbols are often omitted, as is standard.

D. The Computational Diffie-Hellman Assumption

The security of the schemes considered in this work relies on the hardness of the Computational Diffie-Hellman (CDH) problem, of finding g^{ab} given g^a and g^b. We define this assumption more formally, aligning it with principles of concrete security, and first defining the advantage of a constrained adversary in solving CDH.

Definition 1 (CDH advantage)**.** Let G be a cyclic group of finite order n, and g_G be a generator of G. Let $\mathcal{A}$ be a computational Diffie-Hellman adversary that, on input a description of G, n, g_G, and g_G^a and g_G^b for uniformly chosen exponents a and b, returns an element $\mathsf{r} \in G$. The CDH advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathsf{cdh}}_{G,g_G,n}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}}(G, g_G, n) : \mathsf{r} = g_G^{\mathsf{ab}}\right]$$

where the experiment $\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}}(G, g_G, n)$ is defined in Figure 1.

The definition of hardness is natural.

Definition 2 (Hardness of CDH)**.** The CDH problem is said to be (t, ϵ)-hard in G with generator g_G if, for any adversary $\mathcal{A}$ that runs in time at most t, we have

$$\mathsf{Adv}^{\mathsf{cdh}}_{G,g_G,n}(\mathcal{A}) \leq \epsilon$$

We will be more particularly interested in the hardness of the CDH problem in our cyclic group $\mathbb{G}$ of prime order q with generator g, namely $\mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g,q}(\mathcal{A})$.

E. Signature Schemes

Definition 3. A signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ for messages in some set $\mathcal{M}$ consists of three probabilistic algorithms:

- A key generation algorithm KGen that outputs a key pair composed of a public (pk) and private key (sk);
- A signing algorithm Sign that, upon input a private key sk and a message $\mathsf{m} \in \mathcal{M}$, returns a signature σ;
- A verification algorithm Ver that, upon input a public key pk, a message $\mathsf{m} \in \mathcal{M}$ and signature σ, returns a boolean signifying acceptance (with value true, denoted with 1) or rejection (with value false, denoted with 0).

As is standard in cryptography, the desired security property is captured as a game between an adversary attempting to break the security of the scheme (by forging a valid signature) and a challenger that mediates interactions between the adversary and the signature scheme under study to specify adversarial powers and prevent trivial wins.

More specifically, we consider the notion of *existential forgery under adaptive chosen message attacks* (EUF-CMA), in which the adversary—given the public key and oracle access to a signing oracle that signs messages of the adversary's choice under the challenger's key—attempts to produce a valid signature on a *fresh* message—that is, one that has not been queried to the signing oracle. As for CDH, we first define the advantage of an adversary in breaking the security of a signature scheme.

Definition 4 (EUF-CMA advantage)**.** Given a signature scheme $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ and an adversary (or *forger*) $\mathcal{F}$ that, upon input a public key for $\mathcal{S}$ and given oracle access to a signing oracle for $\mathcal{S}$ (which produces a signature upon input a message), the advantage of $\mathcal{A}$ in breaking the EUF-CMA security of $\mathcal{S}$ is

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{S}}(\mathcal{F}) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{S},\mathcal{F}}() : \mathsf{b} \wedge \tilde{\mathsf{m}} \notin \mathcal{Q}_{\mathcal{S}}\right]$$

where the experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{S},\mathcal{F}}()$ is defined in Figure 2, and $\mathcal{Q}_{\mathcal{S}}$ is the set of queries issued by the forger $\mathcal{F}$ to its signing oracle.

We can then naturally define EUF-CMA security as the relevant hardness notion.

[4] In particular, this would normally require the refinement to ensure or check that inputs are indeed valid encodings of elements in the expected algebraic structure, or to prove that it does not matter for security.

```
$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathcal{S},\mathcal{F}}()$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{S}.\mathsf{KGen}()$
$(\tilde{\mathsf{m}},\tilde{\sigma}) \leftarrow \mathcal{F}^{\mathcal{H},\mathcal{G},\mathcal{S}.\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$
$\mathsf{b} \leftarrow \mathcal{S}.\mathsf{Ver}_{\mathsf{pk}}(\tilde{\mathsf{m}},\tilde{\sigma})$
```

Fig. 4. EUF-CMA with two Random Oracles $\mathcal{H}$ and $\mathcal{G}$.

Definition 5 (EUF-CMA security)**.** A signature scheme $\mathcal{S}$ is said to be $(t, q_{\mathcal{S}}, \epsilon)$-*EUF-CMA-secure* if, for any adversary $\mathcal{A}$ that runs in time at most t, making at most $q_{\mathcal{S}}$ queries to its $\mathcal{S}$ oracle, we have

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{S}}(\mathcal{A}) \leq \epsilon$$

F. The Random Oracle Model

The Random Oracle Model (ROM) was introduced by Bellare and Rogaway [11] as a tool to reason about the security of efficient cryptographic constructions that make use of hash functions. In the ROM, hash functions are modelled as *public* oracles that compute a function sampled uniformly from the appropriate function space at the beginning of the experiment. Formally, we equivalently capture random oracles as stateful algorithms whose outputs to *unique* requests are sampled following some distribution, as shown in Figure 3. In the following, we consider uniform random oracles (where d is the uniform distribution over the relevant (finite) output space).

When analysing the security of a scheme that uses hash functions in the ROM, it is crucial to ensure that the adversary is given oracle access also to the hash functions. All the schemes we consider in this paper make use of two distinct hash functions, and we therefore consider forgers who have access to two random oracles $\mathcal{H}$ and $\mathcal{G}$ in addition to the signing oracle. The resulting security experiment is shown in Figure 4. In such a context, the number of queries the forger makes to the random oracles should also be bounded as a resource; in the following we consider a notion of $(t, q_{\mathcal{H}}, q_{\mathcal{G}}, q_{\mathcal{S}}, \epsilon)$-EUF-CMA security naturally extending that from Definitions 4 and 5.

Definition 6 (EUF-CMA security in the ROM)**.** Given two random oracles $\mathcal{H}$ and $\mathcal{G}$ and a signature scheme $\mathcal{S}^{\mathcal{H},\mathcal{G}} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ and an adversary (or *forger*) $\mathcal{F}$ that, upon input a public key for $\mathcal{S}$ and given oracle access to $\mathcal{H}$, $\mathcal{G}$, and to a signing oracle for $\mathcal{S}^{\mathcal{H},\mathcal{G}}$ (which produces a signature upon input a message), the advantage of $\mathcal{A}$ in breaking the EUF-CMA security of $\mathcal{S}$ in the ROM is

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathcal{S}}(\mathcal{F}) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathcal{S},\mathcal{F}}() : \mathsf{b} \wedge \tilde{\mathsf{m}} \notin \mathcal{Q}_{\mathcal{S}}\right]$$

where the experiment $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathcal{S},\mathcal{F}}()$ is that defined in Figure 4, and $\mathcal{Q}_{\mathcal{S}}$ is the set of queries issued by the forger $\mathcal{F}$ to its signing oracle. A signature scheme $\mathcal{S}$ is said to be $(t, q_{\mathcal{H}}, q_{\mathcal{G}}, q_{\mathcal{S}}, \epsilon)$-*EUF-CMA-secure* in the ROM if, for any adversary $\mathcal{A}$ that runs in time at most t, making at most $q_{\mathcal{H}}$ (resp. $q_{\mathcal{G}}$, $q_{\mathcal{S}}$) queries to its $\mathcal{H}$ (resp. $\mathcal{G}$, $\mathcal{S}$) oracle, we have

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathcal{S}}(\mathcal{A}) \leq \epsilon$$

```
$\mathsf{EDL}^{\mathcal{H},\mathcal{G}}_{\mathbb{G},g,q}$

KGen()
  $\mathsf{sk} \leftarrow\!\!\$\; \mathbb{F}_q$
  return $(\mathsf{sk}, g^{\mathsf{sk}})$

Ver_pk(m, (z, r, s, c))
  $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{m},\mathsf{r})$
  $\mathsf{u} \leftarrow g^{\mathsf{s}}\mathsf{pk}^{-\mathsf{c}}$
  $\mathsf{v} \leftarrow \mathsf{h}^{\mathsf{s}}\mathsf{z}^{-\mathsf{c}}$
  $\mathsf{c}' \leftarrow \mathcal{G}(g,\mathsf{h},\mathsf{pk},\mathsf{z},\mathsf{u},\mathsf{v})$
  return $\mathsf{c} = \mathsf{c}'$

Sign_sk(m)
  $\mathsf{r} \leftarrow\!\!\$\; d_{\mathcal{N}}$
  $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{m},\mathsf{r})$
  $\mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}$

  $\mathsf{k} \leftarrow\!\!\$\; \mathbb{F}_q$
  $\mathsf{u} \leftarrow g^{\mathsf{k}}$
  $\mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$
  $\mathsf{c} \leftarrow \mathcal{G}(g,\mathsf{h},g^{\mathsf{sk}},\mathsf{z},\mathsf{u},\mathsf{v})$
  $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c}\cdot\mathsf{sk}$
  return $(\mathsf{z},\mathsf{r},\mathsf{s},\mathsf{c})$
```

Fig. 5. The EDL signature scheme, parameterized by two random oracles $\mathcal{H} : \mathcal{M} \times \mathcal{N} \to \mathbb{G}$ and $\mathcal{G} : \mathbb{G}^6 \to \mathbb{F}_q$.

Although this is not important in this paper, we note in particular the importance—in general—of exposing these two oracles also to any adversary against a construction that *uses* any of the signature schemes we prove secure here. The reductions we formally prove here require *control* over both ROs, which must thus—in general—be independent from any ROs taken over by any further reductions—for example, if the schemes are used in larger protocols.

III. EDL Signatures and their Security

The EDL signature scheme is defined over a set $\mathcal{M}$ of messages as shown in Figure 5, where, in addition to the cyclic group $\mathbb{G}$, we also consider a set $\mathcal{N}$ of *nonces*, equipped with some distribution $d_{\mathcal{N}}$.

Key generation is as standard for DH-style schemes. Signing is done by producing a non-interactive zero-knowledge (NIZK) proof of discrete logarithm equality with bases a message-dependent hash and the generator, and discrete logarithm the secret key. Verification recomputes the message-dependent hash and verifies the corresponding NIZK proof.

We formally prove a *tight* reduction to breaking CDH in $\mathbb{G}$ from breaking EUF-CMA security of EDL, where by tight we mean, like Goh and Jarecki [3] that there are no multiplicative factors involved in relating the time complexity and advantage of an adversary against the scheme and that of the corresponding reduction. More precisely, we are looking for reductions that—when applied to an adversary that breaks the signature scheme with probability ϵ in time t—break the underlying hard problem with probability $\epsilon' \approx \epsilon$ in time $t' \approx t$. Our proof is concrete and constructive: the concrete reduction used in proving Theorem 1 is displayed in Figure 6.

```
$\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}(\mathsf{pk}, \mathsf{g_b})$
var $\mathsf{H} \in \mathcal{M} \times \mathcal{N} \rightharpoonup \mathbb{G} \times \mathbb{F}_q$
var $\mathsf{G} \in \mathbb{G}^6 \rightharpoonup \mathbb{F}_q$

$(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{r}}, \_, \_)) \leftarrow \mathcal{F}^{\mathcal{H},\mathcal{G},\mathsf{Sign}}(\mathsf{pk})$
$\_ \leftarrow \mathcal{H}(\tilde{\mathsf{m}}, \tilde{\mathsf{r}})$ / This is to ensure $(\tilde{\mathsf{m}}, \tilde{\mathsf{r}}) \in \mathsf{H}$
$\mathsf{d} \leftarrow \pi_2(\mathsf{H}[\tilde{\mathsf{m}}, \tilde{\mathsf{r}}])$
return $\tilde{\mathsf{z}} \cdot \mathsf{pk}^{-\mathsf{d}}$

Oracle simulation

$\mathcal{H}(\mathsf{x})$
    $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
    if $(\mathsf{x}) \notin \mathsf{H}$
        $\mathsf{H}[\mathsf{x}] \leftarrow (\mathsf{g_b} g^{\mathsf{d}}, \mathsf{d})$
    return $\pi_1(\mathsf{H}[\mathsf{x}])$

$\mathcal{G}(\mathsf{x})$
    $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
    if $\mathsf{x} \notin \mathsf{G}$
        $\mathsf{G}[\mathsf{x}] \leftarrow c$
    return $\mathsf{G}[\mathsf{x}]$

$\mathsf{Sign}(\mathsf{m})$
    $\mathsf{r} \leftarrow\!\!\$\ d_{\mathcal{N}}$
    $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
    $\mathsf{H}[\mathsf{m}, \mathsf{r}] \leftarrow (g^{\mathsf{d}}, \mathsf{d})$

    $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
    $\mathsf{s} \leftarrow\!\!\$\ \mathbb{F}_q$
    $\mathsf{u} \leftarrow g^{\mathsf{s}} \mathsf{pk}^{-\mathsf{c}}$
    $\mathsf{v} \leftarrow \mathsf{g}^{\mathsf{ds}} \mathsf{pk}^{-\mathsf{dc}}$
    $\mathsf{G}[g, \mathsf{g}^{\mathsf{d}}, \mathsf{pk}, \mathsf{pk}^{\mathsf{d}}, \mathsf{u}, \mathsf{v}] \leftarrow c$
    return $(\mathsf{z}, \mathsf{r}, \mathsf{s}, \mathsf{c})$
```

Fig. 6. The reduction $\mathcal{A}$ from CDH. $\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}$ uses an EUF-CMA forger $\mathcal{F}$ as a black-box, and internally simulates $\mathcal{H}$ and $\mathcal{G}$. H keeps track of both the response h and i. its discrete logarithm in base g (for queries made by the signing oracle), or ii. the unique value $\mathsf{d} \in \mathbb{F}_q$ such that $\mathsf{h} = \mathsf{g_b} g^{\mathsf{d}}$ (for queries made by the forger). π_1 and π_2 are the first and second projections on pairs.

```
$\mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H},\mathcal{G},\mathcal{S},\mathcal{F}}()$
$\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}_q$
$\mathsf{b} \leftarrow\!\!\$\ \mathbb{F}_q$
$\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
$\mathsf{g_b} \leftarrow g^{\mathsf{b}}$
$(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{r}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}})) \leftarrow \mathcal{F}^{\mathcal{H},\mathcal{G},\mathcal{S}}(\mathsf{pk})$
$\mathsf{h} \leftarrow \mathcal{H}(\tilde{\mathsf{m}}, \tilde{\mathsf{r}})$
$\mathsf{u} \leftarrow g^{\tilde{\mathsf{s}}} \mathsf{pk}^{-\tilde{\mathsf{c}}}$
$\mathsf{v} \leftarrow \mathsf{h}^{\tilde{\mathsf{s}}} \mathsf{pk}^{-\tilde{\mathsf{c}}}$
$\mathsf{c} \leftarrow \mathcal{G}(g, \mathsf{h}, \mathsf{pk}, \tilde{\mathsf{z}}, \mathsf{u}, \mathsf{v})$
$\mathsf{win} \leftarrow \tilde{\mathsf{c}} = \mathsf{c} \wedge \tilde{\mathsf{m}} \notin \mathcal{Q}_{\mathcal{S}}$
```

Fig. 7. The EDL proof shim.

Theorem 1 (Security of EDL). *If CDH is (t, ϵ)-hard in $\mathbb{G}$ with generator g, then EDL is $(t', q_{\mathcal{H}}, q_{\mathcal{G}}, q_{\mathcal{S}}, \epsilon')$-EUF-CMA-secure for all non-negative $q_{\mathcal{H}}$, $q_{\mathcal{G}}$, $q_{\mathcal{S}}$, and with*

$$\begin{aligned} t &\lesssim t' + (q_{\mathcal{H}} + 6 \cdot q_{\mathcal{S}} + 1) \cdot t_{exp} \\ \epsilon' &\leq \epsilon + q_{\mathcal{S}} \cdot \left(\frac{q_{\mathcal{H}} + q_{\mathcal{S}}}{|\mathcal{N}|} + \frac{q_{\mathcal{G}} + q_{\mathcal{S}}}{q^2} \right) \\ &\quad + \frac{q_{\mathcal{G}} + 1}{q} \end{aligned}$$

where t_{exp} is the cost of an exponentiation in $\mathbb{G}$.

Proof. The bound on the reduction's time complexity t is not formally verified, and we prove it here: the reduction (as shown in Figure 6) first runs the EUF-CMA forger and simulates its oracles. When simulating each of the forger's $q_{\mathcal{H}}$ queries to $\mathcal{H}$, the reduction computes one exponentiation in $\mathbb{G}$. When simulating each of the forger's $q_{\mathcal{S}}$ queries to the signing oracle, the reduction computes two exponentiations and two double-exponentiations. Finally, the reduction computes one inverse in $\mathbb{F}_q$ (whose cost is omitted, as it is dominated by that of exponentiations) and one exponentiation to retrieve its answer. We overapproximate the cost of double-exponentiation as the cost of two exponentiations to get the bound shown above.

The rest of this Section is dedicated to discussing the proof for the bound on ϵ'. Figures 7-8 show the intermediate games that serve in this proof. □

Goh and Jarecki's original proof [3]—which uses $\mathsf{g_b}^{\mathsf{d}}$ instead of $\mathsf{g_b} g^{\mathsf{d}}$ when simulating $\mathcal{H}$—omits certain low probability cases in which their given simulation fails:

1) when b is 0;
2) when the value of d used in simulating $\mathcal{H}$ for forgery verification is 0; and
3) when the same random nonce is used in two separate signatures of the same message.

Accounting for these cases yields a valid (and indeed machine-checked) security proof, with a slightly worse bound than that given here. We instead formalize and check a different proof, which yields a more precise bound.

A. Intuition

We first note that a valid forgery $(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{r}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}}))$ cannot be such that the signature scheme queried $\mathcal{H}$ on $(\tilde{\mathsf{m}}, \tilde{\mathsf{r}})$ (if that were the case, the forgery would not be fresh), so either the forger has made that query herself, or we can make the query on her behalf after she returns. Note also that, in an honestly computed signature $(\mathsf{z}, \mathsf{r}, \mathsf{s}, \mathsf{c})$ for a message m, we have $\mathsf{h} = \mathsf{z}^{\mathsf{sk}}$, where $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{m}, \mathsf{r})$. The key insight in the reduction is to embed the CDH challenge into the EUF-CMA game, using g^{a} as public key (so that $\mathsf{sk} = \mathsf{a}$), and embedding g^{b} into the answers given by $\mathcal{H}$ to the forger (so that $\mathsf{h} = g^{\mathsf{b}} g^{\mathsf{d}}$ for some d known to the simulator). In this setting, a valid forgery that is such that $\tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}$ can be used to compute g^{ab} as $\tilde{\mathsf{z}} \cdot \mathsf{pk}^{-\mathsf{d}}$.

$$\tilde{\mathsf{z}} \cdot \mathsf{pk}^{-\mathsf{d}} = \mathsf{h}^{\mathsf{a}} \cdot (\mathsf{g}^{\mathsf{a}})^{-\mathsf{d}} = (g^{\mathsf{b}} g^{\mathsf{d}})^{\mathsf{a}} \cdot \mathsf{g}^{-\mathsf{ad}} = g^{\mathsf{ab}}$$

With this key insight, a proof can be obtained by further proving that 1) the reduction—without access to the private key—can simulate the signing oracle an EUF-CMA forger against EDL expects to have access to, and 2) the probability of verification succeeding for a forgery $(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{r}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}}))$ such that $\tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{a}}$ (with $\mathsf{h} \leftarrow \mathcal{H}(\tilde{\mathsf{m}}, \tilde{\mathsf{r}})$) is low.

We now detail the sequence of games and local claims which formalise and leverage this intuition. The sequence of games itself (Lemmas 2 to 4) shows that the simulated oracles from Figure 6 are indistinguishable from the actual EUF-CMA oracles for EDL. We then show (Lemma 6) that any run of the forger that finds a forgery in the final game lets the reduction solve the CDH challenge, except if the forgery is such that $\tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{a}}$, and bound the probability that this occurs with a successful forgery (Lemma 5).

Proofs for the local claims give an intuition of the machine-checked reasoning on programs involved, but do not delve into detailed discussions of the tactics used. We keep the description here as close as possible to the EasyCrypt proof, but omit some formal details in probability-bounding steps. Indeed, in the formal proof, we must first transform the game to explicitly count oracle queries. Our final theorem, like Theorem 1 does quantify over adversaries that make a bounded number of queries.

B. Formalization

We formally define 3 intermediate games that bridge the gap between the $\mathsf{EUF\text{-}CMA}$ experiment and the CDH experiment. We write these intermediate games as instances of a common *shim* $\mathsf{Game}^{\mathsf{EDL}}_{\cdot,\cdot,\cdot,\cdot}()$ (shown in Figure 7), which is parameterized by oracles presenting the same interface as the random and signing oracles, and by a forger. Any two successive games in our formalization differ only in the oracles provided to the shim. The oracles we use in our proof, along with the definition of our intermediate games as instances of the shim, are shown in Figure 8.

Locally, using a shim allows us to focus reasoning on the parts of the experiment that do change between the successive games, and reduces the amount of boilerplate proof code we need to write. In EasyCrypt, we reuse a single proof base over the shim in all proof steps, which reduces the proof obligation to obligations on the oracles themselves, the shim essentially serving as a distinguisher between the various sets of oracles we consider in our games.

The formal proof proceeds in 4 steps:

1) we *refactor* the $\mathsf{EUF\text{-}CMA}$ security of EDL as an instance of the shim;
2) we *embed* the CDH challenge into the responses given to $\mathcal{H}$ queries;
3) we *simulate* the proof of discrete logarithm equality without using the witness (here the secret key); and
4) we show that a forgery either exploits the (negligible) unsoundness of the proof of discrete logarithm equality, or answers the given CDH challenge.

1) Refactoring: The first step refactors the $\mathsf{EUF\text{-}CMA}$ game to make use of the shim, as $\mathsf{Game}^{\mathsf{EDL}}_{0}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H},\mathcal{G},\mathcal{S}_0,\mathcal{F}}()$ where $\mathcal{H}$, $\mathcal{G}$, and $\mathcal{S}_0$ are shown in Figure 8.

Lemma 2 (EDL Refactoring)**.** *For any forger $\mathcal{F}$, we have*

$$\Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{H},\mathcal{G},\mathit{EDL},\mathcal{F}}() : \mathsf{b} \wedge \tilde{\mathsf{m}} \notin \mathcal{Q}_{\mathcal{S}}\right] = \Pr\left[\mathsf{Game}^{\mathit{EDL}}_{0}(\mathcal{F}) : \mathsf{win}\right]$$

Proof. This is a simple program equivalence: $\mathsf{Game}^{\mathsf{EDL}}_{0}(\mathcal{F})$ is the $\mathsf{EUF\text{-}CMA}$ experiment for EDL with KGen and Ver inlined, and with the addition of ineffective operations (sampling of b and computation of $\mathsf{g_b}$, which are not used in the rest of the game. □

2) Embedding: In our second step, we modify the way in which queries to $\mathcal{H}$ are handled to embed the CDH challenge in the log of forger queries. Consider $\mathsf{Game}^{\mathsf{EDL}}_{1}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H}',\mathcal{G},\mathcal{S}_1,\mathcal{F}}()$, with $\mathcal{H}'$, $\mathcal{G}$, $\mathcal{S}_1$ as defined in Figure 8.

To the forger, we expose $\mathcal{H}'$—which computes its output as $\mathsf{g_b}g^{\mathsf{d}}$ for some uniformly sampled d that is kept alongside the oracle's response.

The signing oracle $\mathcal{S}_1$ is modified so that it always samples a fresh value of h (although keeping in its log its discrete logarithm d), regardless of whether its (m, r) had already appeared as a query to $\mathcal{H}'$ or in a signing query. This change is visible to the adversary—who can distinguish between $\mathsf{Game}^{\mathsf{EDL}}_{0}(\mathcal{F})$ and $\mathsf{Game}^{\mathsf{EDL}}_{1}(\mathcal{F})$ when the pair (m, r) being used in a signing query already appears in the map H. We capture this event as $\mathsf{bad}_{\mathcal{H}}$.

In addition, we keep track of an event $\mathsf{bad}_{\mathcal{G}}$, which is triggered when the query to $\mathcal{G}$ made by the signing oracle is not fresh. This event is not observable by the adversary, but is easier to bound in $\mathsf{Game}^{\mathsf{EDL}}_{1}(\mathcal{F})$ than in the next game, where it does become observable.

Lemma 3 (EDL Embedding)**.** *For any forger $\mathcal{F}$, we have*

$$\begin{aligned}\Pr\left[\mathsf{Game}^{\mathit{EDL}}_{0}(\mathcal{F}) : \mathsf{win}\right] &\\ \leq \Pr\left[\mathsf{Game}^{\mathit{EDL}}_{1}(\mathcal{F}) : \mathsf{win}\right]&\\ + \Pr\left[\mathsf{Game}^{\mathit{EDL}}_{1}(\mathcal{F}) : \mathsf{bad}_{\mathcal{H}}\right]&\end{aligned}$$

In addition, for any forger $\mathcal{F}$ that makes at most $q_{\mathcal{H}}$ queries to her $\mathcal{H}$ oracle, and at most $q_{\mathcal{S}}$ queries to her signing oracle, we have

$$\Pr\left[\mathsf{Game}^{\mathit{EDL}}_{1}(\mathcal{F}) : \mathsf{bad}_{\mathcal{H}}\right] \leq q_{\mathcal{S}} \cdot \frac{q_{\mathcal{H}} + q_{\mathcal{S}}}{q}$$

Proof. Thanks to the use of a common shim, we only need to consider differences in behaviour arising from the oracles.

First we observe that logging d in H has no effect on the adversary's view of the system: computing $\mathsf{g_b}g^{\mathsf{d}}$ and g^{d} both yield the uniform distribution over $\mathbb{G}$ when d is uniform in $\mathbb{F}_q$, and the value of d is not used (other than in computing the response of the random oracle) while the forger is running. Focusing on the signing oracle, it is clear that $\mathcal{S}_1$ is equivalent to $\mathcal{S}_0$ as long as $\mathsf{bad}_{\mathcal{H}}$ is false: indeed, in that case, $\mathcal{S}_1$ is exactly $\mathcal{S}_0$ with $\mathcal{H}$ inlined, the conditional reduced to its true branch, and h sampled in a way that reveals its discrete logarithm (this is marked as the dashed box in Figure 8). This proves the first part of the lemma.

Second, we bound the probability of $\mathsf{bad}_{\mathcal{H}}$ occurring. During any one execution of $\mathcal{S}_1$, $\mathsf{bad}_{\mathcal{H}}$ becomes true if a fresh value r sampled in $d_{\mathcal{N}}$ already appears in H. H increases in size by at most 1 for each query to $\mathcal{H}'$ and to $\mathcal{S}_1$. So each query to $\mathcal{S}_1$ sets $\mathsf{bad}_{\mathcal{H}}$ with probability at most $\frac{q_{\mathcal{H}}+q_{\mathcal{S}}}{q}$. A forger that makes $q_{\mathcal{S}}$ query to $\mathcal{S}_1$ therefore triggers $\mathsf{bad}_{\mathcal{H}}$ with probability at most $q_{\mathcal{S}} \cdot \frac{q_{\mathcal{H}}+q_{\mathcal{S}}}{q}$. □

```
$\mathcal{S}_0(\mathsf{m})$
    $\mathsf{r} \leftarrow_{\$} \mathcal{N}$
    $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{m}, \mathsf{r})$
    $\mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}$
    $\mathsf{k} \leftarrow_{\$} \mathbb{F}_q; \mathsf{u} \leftarrow g^{\mathsf{k}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$
    $\mathsf{c} \leftarrow \mathcal{G}(g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
    $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c} \cdot \mathsf{sk}$
    return $(\mathsf{z}, \mathsf{r}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{S}_1(\mathsf{m})$
    $\mathsf{r} \leftarrow_{\$} \mathcal{N}; \mathsf{bad}_{\mathcal{H}} \leftarrow \mathsf{bad}_{\mathcal{H}} \vee (\mathsf{m}, \mathsf{r}) \in \mathsf{H}$
    $\mathsf{d} \leftarrow_{\$} \mathbb{F}_q; \mathsf{h} \leftarrow g^{\mathsf{d}}$
    $\mathsf{H}[\mathsf{m}, \mathsf{r}] \leftarrow (\mathsf{h}, \mathsf{d})$
    $\mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}$
    $\mathsf{k} \leftarrow_{\$} \mathbb{F}_q; \mathsf{u} \leftarrow g^{\mathsf{k}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$
    $\mathsf{bad}_{\mathcal{G}} \leftarrow \mathsf{bad}_{\mathcal{G}} \vee$
            $(g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v}) \in \mathsf{G}$
    $\mathsf{c} \leftarrow \mathcal{G}(g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
    $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c} \cdot \mathsf{sk}$
    return $(\mathsf{z}, \mathsf{r}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{S}_2(\mathsf{m})$
    $\mathsf{r} \leftarrow_{\$} \mathcal{N}$
    $\mathsf{d} \leftarrow_{\$} \mathbb{F}_q; \mathsf{h} \leftarrow g^{\mathsf{d}}$
    $\mathsf{H}[\mathsf{m}, \mathsf{r}] \leftarrow (\mathsf{h}, \mathsf{d})$
    $\mathsf{z} \leftarrow \mathsf{pk}^{\mathsf{d}}$
    $\mathsf{c} \leftarrow_{\$} \mathbb{F}_q; \mathsf{s} \leftarrow_{\$} \mathbb{F}_q$
    $\mathsf{u} \leftarrow g^{\mathsf{s}}\mathsf{pk}^{-\mathsf{c}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{s}}\mathsf{z}^{-\mathsf{c}}$
    $\mathsf{bad}_{\mathcal{G}} \leftarrow \mathsf{bad}_{\mathcal{G}} \vee$
            $(g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v}) \in \mathsf{G}$
    $\mathsf{G}[g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v}] \leftarrow \mathsf{c}$
    return $(\mathsf{z}, \mathsf{r}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{H}(\mathsf{x})$
    $\mathsf{h} \leftarrow_{\$} \mathbb{G}$
    if $\mathsf{x} \notin \mathsf{H}$
        $\mathsf{H}[\mathsf{x}] \leftarrow \mathsf{h}$
    return $\mathsf{H}[\mathsf{x}]$
```

```
$\mathcal{G}(\mathsf{x})$
    $\mathsf{c} \leftarrow_{\$} \mathbb{F}_q$
    if $\mathsf{x} \notin \mathsf{G}$
        $\mathsf{G}[\mathsf{x}] \leftarrow \mathsf{c}$
    return $\mathsf{G}[\mathsf{x}]$
```

```
$\mathcal{H}'(\mathsf{x})$
    $\mathsf{d} \leftarrow_{\$} \mathbb{F}_q$
    if $\mathsf{x} \notin \mathsf{H}$
        $\mathsf{H}[\mathsf{x}] \leftarrow (\mathsf{g_b} g^{\mathsf{d}}, \mathsf{d})$
    return $\pi_1(\mathsf{H}[\mathsf{x}])$
```

Fig. 8. Oracle simulations for use in intermediate steps in the security proof for EDL signatures. Lemmas 2 and 3 use $\mathsf{Game}_0^{\mathsf{EDL}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H},\mathcal{G},\mathcal{S}_0,\mathcal{F}}()$. Lemmas 3 and 4 use $\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H}',\mathcal{G},\mathcal{S}_1,\mathcal{F}}()$. Lemmas 4 and 6 use $\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H}',\mathcal{G},\mathcal{S}_2,\mathcal{F}}()$.

3) Simulation: In our third step, we simulate the proof of discrete logarithm equality. We rely on its zero-knowledge property to simulate the signing oracle without using the secret key.

$\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{EDL}}_{\mathcal{H}',\mathcal{G},\mathcal{S}_2,\mathcal{F}}()$ carries out this simulation by programming the random oracle $\mathcal{G}$ on inputs queried by the signing oracle—regardless of their freshness. $\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F})$ can therefore be distinguished from $\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F})$ by the forger exactly when the signing oracle programs $\mathcal{G}$ in a point the forger had previously queried. This event—which we captured as $\mathsf{bad}_{\mathcal{G}}$ already in $\mathcal{S}_1$—occurs with low probability since some of its parameters are freshly sampled in high entropy distributions. This probability, however, is easier to bound in $\mathcal{S}_1$, and we rely on the argument outlined in Section II-B1 to prove the bound.

Lemma 4 (EDL Simulation)**.** *For any forger $\mathcal{F}$, we have*

$$\begin{aligned}\Pr\Big[\mathsf{Game}_1^{EDL}(\mathcal{F}) : \mathsf{win}\Big] \\ \leq \Pr\Big[\mathsf{Game}_2^{EDL}(\mathcal{F}) : \mathsf{win}\Big] \\ + \Pr\Big[\mathsf{Game}_1^{EDL}(\mathcal{F}) : \mathsf{bad}_{\mathcal{G}}\Big]\end{aligned}$$

In addition, for any forger $\mathcal{F}$ that makes at most $q_{\mathcal{G}}$ queries to her $\mathcal{G}$ oracle, and at most $q_{\mathcal{S}}$ queries to her signing oracle, we have

$$\Pr\Big[\mathsf{Game}_1^{EDL}(\mathcal{F}) : \mathsf{bad}_{\mathcal{G}}\Big] \leq q_{\mathcal{S}} \cdot \frac{q_{\mathcal{G}} + q_{\mathcal{S}}}{q^2}$$

Proof. Note the fact that we bound the probability of $\mathsf{bad}_{\mathcal{G}}$ occurring in the embedding game, rather that the simulation game. This requires a bit of additional effort. We prove the following two relations, combining them to prove the first claim.

$$\Pr\Big[\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{bad}_{\mathcal{G}}\Big] = \Pr\Big[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{bad}_{\mathcal{G}}\Big]$$

$$\begin{aligned}\Pr\Big[\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win}\Big] \\ \leq \Pr\Big[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win}\Big] \\ + \Pr\Big[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{bad}_{\mathcal{G}}\Big]\end{aligned}$$

We do so—both in EasyCrypt and in the argument below—with a single equivalence proof, in which we establish that the $\mathsf{bad}_{\mathcal{G}}$ events occur with the same probability in both games, *and* that a forger $\mathcal{F}$ that wins against $\mathcal{S}_1$ also wins against $\mathcal{S}_2$ unless its run against $\mathcal{S}_2$ triggers $\mathsf{bad}_{\mathcal{G}}$.

The oracles $\mathcal{H}'$ and $\mathcal{G}$ are as in the previous game, with the only difference lying in the signing oracle $\mathcal{S}_2$. We must prove two facts relating executions of $\mathcal{S}_1$ and $\mathcal{S}_2$ starting from the same memory where $\mathsf{bad}_{\mathcal{G}}$ has not yet been set. First, we must prove—unconditionally—that they produce the same distribution over $\mathsf{bad}_{\mathcal{G}}$. Second, we must prove that executions that leave $\mathsf{bad}_{\mathcal{G}}$ unset always yield the same distribution over the output (z, r, s and c) and state (maps H and G) of the oracles.

Variables d and z follow the same distribution in both games, since d is sampled in the same distribution, and $\mathsf{z} = g^{\mathsf{d}\cdot\mathsf{sk}}$.

In order to reason about further equivalences, it is necessary to consider the code of $\mathcal{G}$. Consider a version of $\mathcal{S}_1$ with $\mathcal{G}$ inlined. Now it is clear that variable c can be sampled at the same time as k without change in the semantics. Now, the distribution in $\mathcal{S}_1$ of $(\mathsf{k}, \mathsf{c}, \mathsf{k} + \mathsf{c} \cdot \mathsf{sk})$ is the same as that of $(\mathsf{s} - \mathsf{c} \cdot \mathsf{sk}, \mathsf{c}, \mathsf{s})$ in $\mathcal{S}_2$. The distributions of u, v and $\mathsf{bad}_{\mathcal{G}}$ are therefore also equal. From here on, we only need consider the case where $\mathsf{bad}_{\mathcal{G}}$ does not occur. In this case, the conditional in $\mathcal{G}$ follows the **then** branch, yield the same distribution over output and state. This concludes the proof of the first claim.

Now, as discussed, we bound the probability of $\mathsf{bad}_{\mathcal{G}}$ occurring in $\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F})$ to conclude the proof. Variable $\mathsf{bad}_{\mathcal{G}}$ is set by a signing query if the tuple it uses as input to $\mathcal{G}$ coincides with one of the inputs on which $\mathcal{G}$ has already been queried (one that is set in G). Signing oracle $\mathcal{S}_1$ makes only queries of the form $\left(g, g^{\mathsf{d}}, g^{\mathsf{sk}}, \mathsf{z}, g^{\mathsf{k}}, \mathsf{v}\right)$ where k and d are sampled freshly and uniformly at random in $\mathbb{F}_q$. Therefore, the probability that such a query is one of the at most $q_{\mathcal{G}} + q_{\mathcal{S}}$ already set in G is upper-bounded by $\frac{q_{\mathcal{G}}+q_{\mathcal{S}}}{q^2}$. The forger makes at most $q_{\mathcal{S}}$ queries to its signing oracle, so $\mathsf{bad}_{\mathcal{G}}$ occurs with probability at most $q_{\mathcal{S}} \cdot \frac{q_{\mathcal{G}}+q_{\mathcal{S}}}{q^2}$. □

4) Reduction: In the final step of the proof, we show that if a forger $\mathcal{F}$ wins $\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F})$, the CDH adversary $\mathcal{A}^{\mathcal{F}}$ from Figure 6 succeeds in solving its given CDH instance, except with low probability. More specifically, a successful forgery either can be used to solve the given CDH instance, or relies on an unsound proof of discrete logarithm equality.

We start by stating and proving a lemma bounding the probability of the forger relying on unsoundness.

Lemma 5 ($\tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{sk}}$)**.** *For any forger $\mathcal{F}$ that makes at most $q_{\mathcal{G}}$ queries to her $\mathcal{G}$ oracle, we have*

$$\Pr\left[\mathsf{Game}_2^{EDL}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{sk}}\right] \leq \frac{q_{\mathcal{G}} + 1}{q}$$

Proof. If the forger produces a valid forgery such that $\mathsf{z} \neq \mathsf{h}^{\mathsf{sk}}$ then there exists in map G—at the end of the game's run—an input-output pair $((g, \mathsf{h}, \mathsf{y}, \mathsf{z}, \mathsf{u}, \mathsf{v}), \mathsf{c})$ such that $\mathsf{z} \neq \mathsf{h}^{\log \mathsf{y}}$ and $(\mathsf{u} \times \mathsf{y}^{\mathsf{c}})^{\log \mathsf{h}} = \mathsf{v} \times \mathsf{z}^{\mathsf{c}}$. Indeed, consider the input-output pair $\left(\left(g, \mathsf{h}, g^{\mathsf{sk}}, \tilde{\mathsf{z}}, g^{\tilde{\mathsf{s}}} g^{-\mathsf{sk}\cdot\tilde{\mathsf{c}}}, \mathsf{h}^{\tilde{\mathsf{s}}} g^{-\mathsf{sk}\cdot\tilde{\mathsf{c}}}\right), \mathsf{c}\right)$ involved in the $\mathcal{G}$ query made by the shim during verification of the forgery. Since the forgery is valid, it must be that $\tilde{\mathsf{c}} = \mathsf{c}$ and the conditions hold.

We now bound the probability that any specific query to $\mathcal{G}$ adds such an input-output pair to G. Since all $\mathcal{G}$ queries made by the signing oracle have $\mathsf{z} = \mathsf{h}^{\mathsf{sk}}$, it must be that the $\mathcal{G}$ query we consider was either made by the forger or shim—there are at most $q_{\mathcal{G}} + 1$ such queries. Each of them samples c at random in $\mathbb{F}_q$, and—all inputs being set—there is only one value in $\mathbb{F}_q$ that meets the constraints—namely $\mathsf{c} = {}^{\log(\mathsf{v}/\mathsf{u}^{\log \mathsf{h}})}\!/\!{}_{\log(\mathsf{h}^{\log \mathsf{y}}/\mathsf{z})}$. Oracle $\mathcal{G}$ samples this one value with probability ${}^{1}\!/\!{}_{q}$. □

Lemma 6 (EDL Reduction)**.** *For any forger $\mathcal{F}$ that makes at most $q_{\mathcal{G}}$ queries to her $\mathcal{G}$ oracle, and at most $q_{\mathcal{S}}$ queries to her signing oracle, we have*

$$\Pr\left[\mathsf{Game}_2^{EDL}(\mathcal{F}) : \mathsf{win}\right] \leq \mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g,q}(\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}) + \frac{q_{\mathcal{G}} + 1}{q}$$

Proof. We split the probability depending on whether the forgery breaks the discrete logarithm equality proof's soundness, and bound the summands pairwise to conclude.

$$\begin{aligned}\Pr&\left[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win}\right]\\ &= \Pr\left[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}\right]\\ &\qquad + \Pr\left[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{sk}}\right]\end{aligned}$$

Lemma 5 bounds the second summand. We now bound the first, proving that our reduction can make use of a successful forgery run that does not break the soundness of the proof to solve its CDH challenge.

$$\begin{aligned}\Pr&\left[\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}\right]\\ &\leq \Pr\left[\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}}(\mathbb{G}, g, q) : \mathsf{r} = g^{\mathsf{ab}}\right]\end{aligned}$$

We first show that $\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F})$ and $\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}}(\mathbb{G}, g, q)$—where we recall that $\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}$ is defined in Figure 6—are equivalent as programs until the forger returns (Line 6 in Figure 7). Indeed, note that the shim—up to that point—is an inlined prefix of $\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}}(\mathbb{G}, g, q)$. Its code samples two field elements, hides them in the group, then calls the forger with oracles that are themselves equivalent to those used in $\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}$:

- the simulated $\mathcal{H}$ oracle from Figure 6 is syntactically equal to $\mathcal{H}'$;
- the simulated $\mathcal{G}$ oracle from Figure 6 is syntactically equal to the oracle $\mathcal{G}$ defined in Figure 8; and the signing oracles differ only cosmetically
- we define $\mathsf{h} = g^{\mathsf{d}}$ in $\mathsf{Game}_2^{\mathsf{EDL}}(\mathcal{F})$, and simply propagate the definition in $\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}$; and
- we no longer keep track of $\mathsf{bad}_{\mathcal{G}}$ in the simulated signing oracle.

After the call to $\mathcal{H}$ made by the challenger at Line 6 in Figure 7), we have $\mathsf{H}[\mathsf{x}] = (\mathsf{g_b} g^{\mathsf{d}}, \langle \mathsf{d} \rangle_{\mathcal{A}})$ for some $\mathsf{d} \in \mathbb{F}_q$. In this case, we know that $\mathsf{h} = \mathsf{g_b} g^{\mathsf{d}}$ and the CDH adversary recovers the value d corresponding to h from the random oracle's record. We, therefore always have $\tilde{\mathsf{z}} \times \mathsf{pk}^{-\mathsf{d}} = \mathsf{g_b}^{\mathsf{sk}} g^{\mathsf{d}\cdot\mathsf{sk}} g^{-\mathsf{d}\cdot\mathsf{sk}} = \mathsf{g_b}^{\mathsf{sk}}$. Our reduction runs with $\mathsf{sk} = \mathsf{a}$ and $\mathsf{g_b} = g^{\mathsf{b}}$, where a and b are the CDH secrets. Therefore, if the forger wins with a valid forgery such that $\tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}$, then the reduction returns the value of g^{ab}. □

Combining Lemmas 2, 3, 4, and 6 allows us to conclude the proof of Theorem 1.

```
$\mathsf{CM}^{\mathcal{H},\mathcal{G}}_{\mathbb{G},g,q}$

$\mathsf{KGen}()$
    $\mathsf{sk} \leftarrow_{\$} \mathbb{F}_q$
    return $(\mathsf{sk}, g^{\mathsf{sk}})$

$\mathsf{Ver}_{\mathsf{pk}}(\mathsf{m}, (\mathsf{z}, \mathsf{s}, \mathsf{c}))$
    $\mathsf{u} \leftarrow g^{\mathsf{s}}\mathsf{pk}^{-\mathsf{c}}$
    $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{u})$
    $\mathsf{v} \leftarrow \mathsf{h}^{\mathsf{s}}\mathsf{z}^{-\mathsf{c}}$
    $\mathsf{c}' \leftarrow \mathcal{G}(\mathsf{m}, g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
    return $\mathsf{c} = \mathsf{c}'$

$\mathsf{Sign}_{\mathsf{sk}}(\mathsf{m})$
    $\mathsf{k} \leftarrow_{\$} \mathbb{F}_q$
    $\mathsf{u} \leftarrow g^{\mathsf{k}}$
    $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{u})$
    $\mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}$
    $\mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$

    $\mathsf{c} \leftarrow \mathcal{G}(\mathsf{m}, g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
    $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c} \cdot \mathsf{sk}$
    return $(\mathsf{z}, \mathsf{s}, \mathsf{c})$
```

Fig. 9. The CM signature scheme, parameterized by two random oracles $\mathcal{H} : \mathbb{G} \to \mathbb{G}$ and $\mathcal{G} : \mathcal{M} \times \mathbb{G}^6 \to \mathbb{F}_q$.

IV. CM Signatures and their Security

The CM signature scheme is defined over messages in $\mathcal{M}$ as shown in Figure 9. The most notable difference from EDL is that the message is not included in the random oracle query to $\mathcal{H}$ whose output serves as the second base for the proof of discrete logarithm equality; instead, the message is included as an auxiliary input to the challenge-generating random oracle query, in a way similar to standard Schnorr signatures. This change supports security without the use of additional randomness (which shortens the signature), but also allows the use of coupons, where the most part of the signature's computation can be done offline and ahead of the message being produced. The on-line part of the signature then simply consists in one hash query, and two simple field arithmetic operations.

Following Chevallier-Mames [5], we formally prove a *tight* reduction to breaking CDH in $\mathbb{G}$ from breaking the EUF-CMA security of CM. The proof is, here again, concrete and constructive. The reduction is displayed in Figure 10. Unlike the EDL simulator, the simulator shown in Figure 10 must keep track of different information when simulating random oracle queries placed by the forger or internal to the simulation of signatures. We capture this in code using a tagged disjoint union, denoting with $\langle X \rangle_{t_\ell} \uplus \langle Y \rangle_{t_r}$ the set that contains values from set X tagged with t_l and values from set Y tagged with t_r (for distinct tags). We use a tagged tuple notation (with $\langle v \rangle_t$ denoting a value v tagged with tag t) to denote values in sum types, using **match** to match on the tag t and bind the tuple's elements to names provided in the pattern. We later use π as a notation to project out a tagged union's contents so that $\pi(\langle v \rangle_t) = v$.

Theorem 7 (Security of CM)**.** *If CDH is (t, ϵ)-hard in $\mathbb{G}$ with generator g, then CM is $(t', q_{\mathcal{H}}, q_{\mathcal{G}}, q_{\mathcal{S}}, \epsilon')$-EUF-CMA-secure for all non-negative $q_{\mathcal{H}}$, $q_{\mathcal{G}}$, $q_{\mathcal{S}}$ and with*

$$\begin{aligned} t &\lesssim t' + (6 \cdot q_{\mathcal{S}} + (q_{\mathcal{H}} + 1) + 5) \cdot t_{exp} \\ \epsilon' &\leq \epsilon + q_{\mathcal{S}} \cdot \left(\frac{q_{\mathcal{H}} + q_{\mathcal{S}}}{q} + \frac{q_{\mathcal{G}} + q_{\mathcal{S}}}{q^2} \right) \\ &\quad + \frac{q_{\mathcal{G}} + 1}{q} + 2 \cdot \frac{q_{\mathcal{S}} \cdot (q_{\mathcal{G}} + 1)}{q} \end{aligned}$$

where t_{exp} is the cost of an exponentiation in $\mathbb{G}$.

Proof. The complexity bound is not formally verified, so we argue the time bound here: the reduction (see Figure 10) first runs the EUF-CMA forger, simulating its oracles. When simulating $\mathcal{H}$, the reduction computes one exponentiation and one product in $\mathbb{G}$ (whose cost we omit). Simulating $\mathcal{G}$ is straightforward and incurs no cost. Simulating signature queries requires two exponentiations and two double exponentiations (which we cost as four exponentiations). Finally, extracting the CDH solution from the simulator's state and the forger's output costs one $\mathcal{H}$ query, two double exponentiations and one exponentiation (with some operations on exponents whose cost is omitted, as dominated by the cost of exponentations in $\mathbb{G}$).

The rest of this Section is dedicated to discussing the proof. As with EDL, we include a high-level and intuitive overview and details of its formalization, but do so at a lesser level of detail, focusing instead of aspects that differ, and patterns that are common to both proofs. □

A. Proof Overview

The security argument for CM signatures is slightly less intuitive. Indeed, the statement being proved as part of the signing process is entirely independent from the message which is instead included into the challenge. This means that valid forgeries can in fact involve a query to $\mathcal{H}$ that was made by the signing oracle.

If the forger produces a forgery such that $\mathsf{z} \neq \mathsf{h}^{\mathsf{sk}}$, or in cases where the verification of the forgery involves a fresh query to $\mathcal{H}$, or one that was made by the forger, the proof is exactly as that of EDL. However, there is one additional case to consider. Indeed, the final reduction for the EDL scheme exploits the fact that the message to be signed is included in the input to $\mathcal{H}$ to deduce that a fresh forgery *must* involve a query to $\mathcal{H}$ whose response contains the CDH challenge g^{b}.

In CM, we cannot exclude a valid signature $(\tilde{\mathsf{z}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}})$ on some fresh message $\tilde{\mathsf{m}}$ where $\mathsf{u} = g^{\tilde{\mathsf{s}}}\mathsf{pk}^{-\tilde{\mathsf{c}}}$ was previously used as input to $\mathcal{H}$ in a signing query. In such a situation, however, we have two *valid* pairs (c, s) and $(\tilde{\mathsf{c}}, \tilde{\mathsf{s}})$ such that $g^{\mathsf{s}}\mathsf{pk}^{-\mathsf{c}} = \mathsf{u} = g^{\tilde{\mathsf{s}}}\mathsf{pk}^{-\tilde{\mathsf{c}}}$ (the former from the simulator's log of the original query of $\mathcal{H}$ on u, and the latter from the forgery itself), and we can recover the secret key as $\mathsf{sk} = \frac{\mathsf{s} - \tilde{\mathsf{s}}}{\mathsf{c} - \tilde{\mathsf{c}}}$ when $\mathsf{c} \neq \tilde{\mathsf{c}}$. There is a low probability of having $\mathsf{c} = \tilde{\mathsf{c}}$ here, which is accounted for using a final failure event col, which captures collisions in the output of $\mathcal{G}$, and whose probability accounts for the final term in the probability bound. (We in fact need to capture only *some* such collisions, since we also know that the forgery is fresh. Details are discussed in relation to Lemma 12, below.) As such, and interestingly, the

```
$\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}(\mathsf{pk}, \mathsf{g_b})$
var $\mathsf{H} \in \mathbb{G} \rightharpoonup \mathbb{G} \times (\langle \mathbb{F}_q \rangle_{\mathcal{A}} \uplus \langle \mathbb{F}_q \times \mathbb{F}_q \rangle_{\mathcal{S}})$
var $\mathsf{G} \in \mathcal{M} \times \mathbb{G}^6 \rightharpoonup \mathbb{F}_q$

$(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}})) \leftarrow \mathcal{F}^{\mathcal{H},\mathcal{G},\mathsf{Sign}}(\mathsf{pk})$
$\mathsf{u} \leftarrow g^{\tilde{\mathsf{s}}} \mathsf{pk}^{-\tilde{\mathsf{c}}}$
$\_ \leftarrow \mathcal{H}(\mathsf{u})$ / This is to ensure $\mathsf{u} \in H$
match $(\pi_2(\mathsf{H}[\mathsf{u}]))$ with
| $\langle \mathsf{d} \rangle_{\mathcal{A}} \mapsto$ return $\tilde{\mathsf{z}} \cdot \mathsf{pk}^{-\mathsf{d}}$
| $\langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}} \mapsto$ return $\mathsf{g_b}^{\frac{\mathsf{s}-\tilde{\mathsf{s}}}{\mathsf{c}-\tilde{\mathsf{c}}}}$

Oracle simulation

$\mathcal{H}(\mathsf{u})$
    $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
    if $\mathsf{u} \notin \mathsf{H}$
        $\mathsf{H}[\mathsf{u}] \leftarrow (\mathsf{g_b} \cdot g^{\mathsf{d}}, \langle \mathsf{d} \rangle_{\mathcal{A}})$
    return $\pi_1(\mathsf{H}[\mathsf{u}])$

$\mathcal{G}(\mathsf{x})$
    $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
    if $\mathsf{x} \notin \mathsf{G}$
        $\mathsf{G}[\mathsf{x}] \leftarrow \mathsf{c}$
    return $\mathsf{G}[\mathsf{x}]$

$\mathsf{Sign}(\mathsf{m})$
    $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
    $\mathsf{s} \leftarrow\!\!\$\ \mathbb{F}_q$
    $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$

    $\mathsf{u} \leftarrow g^{\mathsf{s}} \mathsf{pk}^{-\mathsf{c}}; \mathsf{h} \leftarrow g^{\mathsf{d}}$
    $\mathsf{z} \leftarrow \mathsf{pk}^{\mathsf{d}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{s}} \mathsf{z}^{-\mathsf{c}}$

    $\mathsf{H}[\mathsf{u}] \leftarrow (\mathsf{h}, \langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}})$
    $\mathsf{G}[\mathsf{m}, g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v}] \leftarrow \mathsf{c}$
    return $(\mathsf{z}, \mathsf{s}, \mathsf{c})$
```

Fig. 10. The reduction $\mathcal{A}$ to CDH. $\mathcal{A}$ uses an EUF-CMA forger $\mathcal{F}$ as a black-box, and internally simulates $\mathcal{H}$ and $\mathcal{G}$ through initially empty finite maps H and G. H keeps track of both the response h and the random exponents s and c used in the related signature (for queries made by the signing oracle), or the value $\log_g(\mathsf{h} \cdot \mathsf{g_b}^{-1})$ (for direct queries). π_1 and π_2 are the first and second projections on pairs.

```
$\mathsf{Game}^{\mathsf{CM}}_{\mathcal{H},\mathcal{G},\mathcal{S},\mathcal{F}}()$
$\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}_q$
$\mathsf{b} \leftarrow\!\!\$\ \mathbb{F}_q$
$\mathsf{pk} \leftarrow g^{\mathsf{sk}}$
$\mathsf{g_b} \leftarrow g^{\mathsf{b}}$
  $(\tilde{\mathsf{m}}, (\tilde{\mathsf{z}}, \tilde{\mathsf{s}}, \tilde{\mathsf{c}})) \leftarrow \mathcal{F}^{\mathcal{H},\mathcal{G},\mathcal{S}}(\mathsf{pk})$
  $\mathsf{u} \leftarrow g^{\mathsf{s}} \mathsf{pk}^{-\tilde{\mathsf{c}}}$
  $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{u})$
$\mathsf{v} \leftarrow \mathsf{h}^{\tilde{\mathsf{s}}} \mathsf{pk}^{-\tilde{\mathsf{c}}}$
$\mathsf{c} \leftarrow \mathcal{G}(\tilde{\mathsf{m}}, g, \mathsf{h}, \mathsf{pk}, \tilde{\mathsf{z}}, \mathsf{u}, \mathsf{v})$
$\mathsf{win} \leftarrow \tilde{\mathsf{c}} = \mathsf{c} \wedge \tilde{\mathsf{m}} \notin \mathcal{Q}_{\mathcal{S}}$
```

Fig. 11. The $\mathsf{Game}^{\mathsf{CM}}_{\cdot,\cdot,\cdot,\cdot}()$ shim used in the security proof for CM. We use a dashed box to isolate code that serves as the core of the reduction (Figure 10).

proof still leverages the special soundness of Schnorr proofs, but does so *without* making use of the forking lemma. The scheme itself is designed so that the forger gives the simulator (via random oracle queries) enough information to fork its execution without rewinding when producing a forgery from which a CDH solution can only be extracted via special soundness.

B. Formalization

As before, throughout the formal proof, we make use of a shim game (Figure 11), which remains unchanged except in the first and last steps, allowing us to focus the formal reasoning (and, here, the discussions) on meaningful changes in the oracles and to limit boilerplate proof artefacts. As before, variables shared between the shim and the intermediate oracle definitions in Figure 12 are simply made global in the shim so they can be accessed directly.

Figure 12 shows the oracles we use in the sequence of games, with claims on the game transitions displayed as Lemmas 8 and 9, and on the final reduction as Lemma 12. In our CM sequence of games, we omit details of the refactoring step, but display the corresponding oracles (in Figure 12) for completeness and clarity in proofs.

1) Step 1 — Embedding: In this proof step (Lemma 8), we change the way in which queries to $\mathcal{H}$ are handled. We expose $\mathcal{H}'$ to the forger so that $\mathsf{g_b}$ can easily be recovered by the simulator from the answers given, without affecting their distribution. In the signing oracle, we program answers to internal $\mathcal{H}'$ queries regardless of previous queries. The corresponding simulations are shown in Figure 12.

As with the EDL proof, oracle $\mathcal{S}_1$ keeps track of a failure event $\mathsf{bad}_{\mathcal{G}}$ that will only become relevant in Lemma 9.

Lemma 8 (CM Embedding)**.** *For all forgers $\mathcal{F}$ that make at most $q_{\mathcal{H}}$ queries to their $\mathcal{H}$ oracle, and at most $q_{\mathcal{S}}$ queries to their signing oracles, we have*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathit{CM}}(\mathcal{F}) \leq \Pr\left[\mathsf{Game}^{\mathit{CM}}_1(\mathcal{F}) : \mathsf{win}\right] + q_{\mathcal{S}} \cdot \frac{q_{\mathcal{H}} + q_{\mathcal{S}}}{q}$$

Proof. The proof is an easy replay of those of Lemma 2 and Lemma 3, noting that the randomiser in the input to $\mathcal{H}$ is now sampled in $\mathbb{G}$ instead of $\mathcal{N}$. □

2) Step 2 — Simulation: The second step (Lemma 9) simulates the zero-knowledge proof by programming the relevant random oracle in the signing oracle. In this proof, unlike in that for EDL, we need to modify also the random oracle exposed to the forger as $\mathcal{G}'$ to detect collisions in $\mathcal{G}'$ as they happen, and to keep track of which of the forger or signing oracle made particular queries (by internally tagging responses). We wield both of these tools in the proof of Lemma 12, below.

Lemma 9 (CM Simulation)**.** *For all forgers $\mathcal{F}$ that make at most $q_{\mathcal{G}}$ queries to their $\mathcal{G}$ oracle and at most $q_{\mathcal{S}}$ queries to their signing oracle, we have*

$$\Pr\left[\mathsf{Game}^{\mathit{CM}}_1(\mathcal{F}) : \mathsf{win}\right] \leq \Pr\left[\mathsf{Game}^{\mathit{CM}}_2(\mathcal{F}) : \mathsf{win}\right] + q_{\mathcal{S}} \cdot \frac{q_{\mathcal{G}} + q_{\mathcal{S}}}{q^2}$$

Proof. The proof is the same as that of Lemma 4, noting in addition that the $\mathsf{col}_{\mathcal{S}}$ and $\mathsf{col}_{\mathcal{A}}$ events do not affect the oracles' behaviour, and that neither does internally tagging responses from $\mathcal{G}'$ with the party that first observed them. □

```
$\mathcal{S}_0(\mathsf{m})$
  $\mathsf{k} \leftarrow\!\!\$\ \mathbb{F}_q$
  $\mathsf{u} \leftarrow g^{\mathsf{k}}$
  $\mathsf{h} \leftarrow \mathcal{H}(\mathsf{u})$
  $\mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$
  $\mathsf{c} \leftarrow \mathcal{G}(\mathsf{m}, g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
  $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c} \cdot \mathsf{sk}$
  return $(\mathsf{z}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{H}(\mathsf{u})$
  $\mathsf{h} \leftarrow\!\!\$\ \mathbb{G}$
  if $\mathsf{u} \notin \mathsf{H}$
    $\mathsf{H}[\mathsf{u}] \leftarrow \mathsf{h}$
  return $\mathsf{H}[\mathsf{u}]$
```

```
$\mathcal{G}(\mathsf{x})$
  $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
  if $\mathsf{x} \notin \mathsf{G}$
    $\mathsf{G}[\mathsf{x}] \leftarrow \mathsf{c}$
  return $\mathsf{G}[\mathsf{x}]$
```

```
$\mathcal{S}_1(\mathsf{m})$
  $\mathsf{k} \leftarrow\!\!\$\ \mathbb{F}_q$
  $\mathsf{u} \leftarrow g^{\mathsf{k}}; \mathsf{bad}_{\mathcal{H}} \leftarrow \mathsf{bad}_{\mathcal{H}} \vee \mathsf{u} \in \mathsf{H}$
  $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
  $\mathsf{h} \leftarrow g^{\mathsf{d}}; \mathsf{z} \leftarrow \mathsf{h}^{\mathsf{sk}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{k}}$
  $\mathsf{bad}_{\mathcal{G}} \leftarrow \mathsf{bad}_{\mathcal{G}} \vee$
      $(\mathsf{m}, g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v}) \in \mathsf{G}$
  $\mathsf{c} \leftarrow \mathcal{G}(\mathsf{m}, g, \mathsf{h}, g^{\mathsf{sk}}, \mathsf{z}, \mathsf{u}, \mathsf{v})$
  $\mathsf{s} \leftarrow \mathsf{k} + \mathsf{c} \cdot \mathsf{sk}$
  $\mathsf{H}[\mathsf{u}] \leftarrow (\mathsf{h}, \langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}})$
  return $(\mathsf{z}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{H}'(\mathsf{u})$
  $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
  if $\mathsf{u} \notin \mathsf{H}$
    $\mathsf{H}[\mathsf{u}] \leftarrow (\mathsf{g_b} g^{\mathsf{d}}, \langle \mathsf{d} \rangle_{\mathcal{A}})$
  return $\pi_1(\mathsf{H}[\mathsf{u}])$
```

```
$\mathcal{S}_2(\mathsf{m})$
  $\mathsf{s} \leftarrow\!\!\$\ \mathbb{F}_q; \mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
  $\mathsf{u} \leftarrow g^{\mathsf{s}} \mathsf{pk}^{-\mathsf{c}}$
  $\mathsf{d} \leftarrow\!\!\$\ \mathbb{F}_q$
  $\mathsf{h} \leftarrow g^{\mathsf{d}}; \mathsf{z} \leftarrow \mathsf{pk}^{\mathsf{d}}; \mathsf{v} \leftarrow \mathsf{h}^{\mathsf{s}} \mathsf{z}^{-\mathsf{c}}$
  $\mathsf{bad}_{\mathcal{G}} \leftarrow \mathsf{bad}_{\mathcal{G}} \vee$
      $(\mathsf{m}, g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v}) \in \mathsf{G}$
  $\mathsf{col}_{\mathcal{S}} \leftarrow \mathsf{col}_{\mathcal{S}} \vee \exists \tilde{x}.\ \mathsf{G}[\tilde{x}] = \langle \mathsf{c} \rangle_{\mathcal{A}}$
  $\mathsf{G}[\mathsf{m}, g, \mathsf{h}, \mathsf{pk}, \mathsf{z}, \mathsf{u}, \mathsf{v}] \leftarrow \langle \mathsf{c} \rangle_{\mathcal{S}}$
  $\mathsf{H}[\mathsf{u}] \leftarrow (\mathsf{h}, \langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}})$
  return $(\mathsf{z}, \mathsf{s}, \mathsf{c})$
```

```
$\mathcal{G}'(\mathsf{x})$
  $\mathsf{c} \leftarrow\!\!\$\ \mathbb{F}_q$
  if $\mathsf{x} \notin \mathsf{G}$
    $\mathsf{col}_{\mathcal{A}} \leftarrow \mathsf{col}_{\mathcal{A}} \vee \exists \tilde{x}.\ \mathsf{G}[\tilde{x}] = \langle \mathsf{c} \rangle_{\mathcal{S}}$
    $\mathsf{G}[\mathsf{x}] \leftarrow \langle \mathsf{c} \rangle_{\mathcal{A}}$
  return $\pi(\mathsf{G}[\mathsf{x}])$
```

Fig. 12. Oracle simulations for use in intermediate steps in the security proof for Chevallier-Mames signatures. Lemmas 8 and 9 use $\mathsf{Game}_1^{\mathsf{CM}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{CM}}_{\mathcal{H}',\mathcal{G},\mathcal{S}_1,\mathcal{F}}()$. Lemmas 9 and 12 use $\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) := \mathsf{Game}^{\mathsf{CM}}_{\mathcal{H}',\mathcal{G}',\mathcal{S}_2,\mathcal{F}}()$.

3) Step 3 — Reduction: Finally, the reduction step shows that if a forger $\mathcal{F}$ wins $\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F})$, the CDH adversary $\mathcal{A}^{\mathcal{F}}$ from Figure 10 succeeds in solving its given CDH instance, except with low probability. We prove (Lemma 12) that a successful forgery: i. solves the given CDH instance; ii. relies on an unsound proof of discrete logarithm equality (Lemma 10); or iii. finds a (restricted) collision in the random oracle (Lemma 11).

We start by bounding the two events that prevent the simulator from solving CDH.

Lemma 10. *For all forgers $\mathcal{F}$ that make at most $q_{\mathcal{G}}$ queries to their $\mathcal{G}$ oracle, we have*

$$\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{sk}}\right] \leq \frac{q_{\mathcal{G}} + 1}{q}$$

Proof. The proof is identical to that of EDL (Lemma 5). □

Lemma 11. *For all forgers $\mathcal{F}$ that make at most $q_{\mathcal{G}}$ queries to their $\mathcal{G}$ oracle and at most $q_{\mathcal{S}}$ queries to their signing oracle, we have*

$$\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{A}} \vee \mathsf{col}_{\mathcal{S}}\right] \leq 2 \cdot \frac{q_{\mathcal{S}} \cdot (q_{\mathcal{G}} + 1)}{q}$$

Proof. First note that

$$\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{A}} \vee \mathsf{col}_{\mathcal{S}}\right]$$
$$\leq \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{A}}\right] + \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{S}}\right]$$

We now show that $\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{A}}\right] \leq \frac{q_{\mathcal{S}} \cdot (q_{\mathcal{G}}+1)}{q}$.

The $\mathsf{col}_{\mathcal{A}}$ event occurs when the freshly sampled value c coincides with one of the c values previously sampled by the signing oracle. There are at most $q_{\mathcal{S}}$ such values, and the event occurs with probability at most $\frac{q_{\mathcal{S}}}{q}$ during each one of the at most $q_{\mathcal{G}} + 1$ queries made to $q_{\mathcal{G}}$ during the execution of $\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F})$ (at most $q_{\mathcal{S}}$ made by the forger, and one made to validate the forgery).

To conclude the proof, we establish the same bound on $\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{S}}\right]$ using a symmetric argument (with the roles of $q_{\mathcal{G}} + 1$ and $q_{\mathcal{S}}$ reversed). □

Lemma 12 (CM Reduction)**.** *For all forgers $\mathcal{F}$ that make at most $q_{\mathcal{G}}$ queries to their $\mathcal{G}$ oracle, and at most $q_{\mathcal{S}}$ queries to their signing oracle, we have*

$$\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win}\right]$$
$$\leq \mathsf{Adv}^{\mathsf{cdh}}_{\mathbb{G},g,q}(\mathcal{A}^{\mathcal{F}}) + \frac{q_{\mathcal{G}} + 1}{q} + 2 \cdot \frac{q_{\mathcal{S}} \cdot (q_{\mathcal{G}} + 1)}{q}$$

Proof. First, note that we can split the forger's success probability as follows, for any forger $\mathcal{F}$.

$$\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win}\right]$$
$$= \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}\right]$$
$$+ \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \wedge \tilde{\mathsf{z}} \neq \mathsf{h}^{\mathsf{sk}}\right]$$

We can further split the first summand based on whether one of the collision events captured as $\mathsf{col}_{\mathcal{A}}$ and $\mathsf{col}_{\mathcal{S}}$ occurred during the run.

$$\begin{aligned}\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \land \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}}\right] \\ \leq \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{col}_{\mathcal{A}} \lor \mathsf{col}_{\mathcal{S}}\right] \\ + \Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \land \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}} \land \neg(\mathsf{col}_{\mathcal{A}} \lor \mathsf{col}_{\mathcal{S}})\right]\end{aligned}$$

With Lemmas 10 and 11, it is now sufficient to prove that a forger that wins with a forgery such that $\mathsf{z} = \mathsf{h}^{\mathsf{sk}}$ and without causing a collision in $\mathcal{G}$ allows our reduction to solve its given CDH instance.

$$\begin{aligned}\Pr\left[\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F}) : \mathsf{win} \land \tilde{\mathsf{z}} = \mathsf{h}^{\mathsf{sk}} \land \neg(\mathsf{col}_{\mathcal{A}} \lor \mathsf{col}_{\mathcal{S}})\right] \\ \leq \Pr\left[\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}}(\mathbb{G}, g, q) : r = g^{ab}\right]\end{aligned}$$

First observe that $\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F})$ and $\mathsf{Exp}^{\mathsf{cdh}}_{\mathcal{A}}(\mathbb{G}, g, q)$, as programs, share i. their oracles (up to projection on $\mathcal{G}$, and the removal of code that tracks failure events $\mathsf{bad}_{\mathcal{G}}$, $\mathsf{col}_{\mathcal{S}}$ and $\mathsf{col}_{\mathcal{A}}$); and ii. a common prefix, which ends at the end of the dashed box shown in Figure 11 (for $\mathsf{Game}_2^{\mathsf{CM}}(\mathcal{F})$) and after the third line of $\mathcal{A}^{\mathcal{F}}_{\mathbb{G},g,q}$ (for the CDH game). Up until those points, the two programs are perfectly equivalent, and produce the same state (up to projections on G).

Consider two possible cases after the call to $\mathcal{H}$ made by the challenger as part of the verification query (or, rather, at the end of the dashed box in Figure 11): either i. $\mathsf{H}[\mathsf{u}] = (\mathsf{g_b} g^{\mathsf{d}}, \langle \mathsf{d} \rangle_{\mathcal{A}})$ for some d (when u was not used in a signature); here, the same proof as in EDL applies, or ii. $\mathsf{H}[\mathsf{u}] = (g^{\mathsf{d}}, \langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}})$ for some d, s and c (when u was used in a signature).

In the second case, we have $\mathsf{H}[\mathsf{u}] = (g^{\mathsf{d}}, \langle \mathsf{s}, \mathsf{c} \rangle_{\mathcal{S}})$ for some d, s and c such that $g^{\mathsf{s} - \mathsf{sk} \cdot \mathsf{c}} = \mathsf{u} = g^{\tilde{\mathsf{s}} - \mathsf{sk} \cdot \tilde{\mathsf{c}}}$ (where the first equality comes from the fact that the signing oracle only inserts pairs of values with that property into H, and the second equality comes from the construction of u by the reduction). If $\mathsf{c} \neq \tilde{\mathsf{c}}$, then we can compute the discrete logarithm $\mathsf{sk} = \frac{\mathsf{s} - \tilde{\mathsf{s}}}{\mathsf{c} - \tilde{\mathsf{c}}}$ of pk, allowing us to solve the CDH instance $(\mathsf{pk}, \mathsf{g_b})$ given as input to $\mathcal{A}$—simply by computing $\mathsf{g_b}^{\mathsf{sk}}$ ($\mathsf{sk} = \mathsf{a}$ and $\mathsf{g_b} = g^{\mathsf{b}}$, where a and b are the CDH secrets).

It remains to show that we are in a case where $\mathsf{c} \neq \tilde{\mathsf{c}}$. We know that u was queried to $\mathcal{H}$ by the signing oracle. Therefore, it must be that there exists a message m such that $\mathsf{G}[\mathsf{m}, g, \mathsf{h}, \mathsf{pk}, \mathsf{h}^{\mathsf{sk}}, \mathsf{u}, \mathsf{u}^{\mathsf{sk}}] = \langle \mathsf{c} \rangle_{\mathcal{S}}$ (since the signing oracle always inserts such an element in G after inserting an element in H). Since the forgery is fresh, it must be that $\mathsf{m} \neq \tilde{\mathsf{m}}$. Since the forgery is also valid, it must also be that $\mathsf{G}[\tilde{\mathsf{m}}, g, \mathsf{h}, \mathsf{pk}, \mathsf{h}^{\mathsf{sk}}, \mathsf{u}, \mathsf{u}^{\mathsf{sk}}] = \langle \tilde{\mathsf{c}} \rangle_{\mathcal{A}}$ (the query may be fresh; this does not impact the reasoning). Therefore, it must be that $\mathsf{c} \neq \tilde{\mathsf{c}}$ or one of $\mathsf{col}_{\mathcal{A}}$ or $\mathsf{col}_{\mathcal{S}}$ would have occurred. □

As before, combining Lemmas 8, 9 and 12 allows us to conclude the proof of Theorem 7.

V. Conclusion

We now briefly discuss and reflect on the formal development itself before discussing related work (both on tight DL-based signatures and machine-checked cryptographic proofs) and highlighting interesting directions for further generalizations beyond the shim presented here and beyond simple digital signatures.

A. Formal Development

We now discuss some aspects of the formal development.

Differences between paper and formal proofs: The bounds we prove formally are slightly tighter than those presented in this paper.

In practice, our formal proof bounds the probability of $\mathsf{bad}_{\mathcal{H}}$ as $\Pr\left[\mathsf{Game}_1^{\mathsf{EDL}}(\mathcal{F}) : \mathsf{bad}_{\mathcal{H}}\right] \leq q_{\mathcal{S}} \cdot \frac{q_{\mathcal{H}} + \frac{q_{\mathcal{S}} - 1}{2}}{q}$ instead of $q_{\mathcal{S}} \cdot \frac{q_{\mathcal{H}} + q_{\mathcal{S}}}{q}$ for EDL, and has similarly tighter bounds for $\mathsf{bad}_{\mathcal{G}}$ and for both events in the CM proof. This is simply due to more precise accounting of queries: during the ith query to $\mathcal{S}$, we can bound the number of entries in H by $q_{\mathcal{H}} + i - 1$ instead of $q_{\mathcal{H}} + q_{\mathcal{S}}$, and the bound is $\sum_{0 \leq i < q_{\mathcal{S}}} q_{\mathcal{H}} + i$ instead of the bound $\sum_{0 \leq i < q_{\mathcal{S}}} q_{\mathcal{H}} + q_{\mathcal{S}}$ discussed in the proofs above. We choose to keep the bound (and related discussions) simple in this presentation, as these details add little to the paper's contributions while making the pen-and-paper argument more difficult to trust.

In order to formally obtain the tight bounds discussed here, our formal shim is slightly more complex than the ones shown in Figures 7 and 11, and takes two different $\mathcal{G}$ oracles, with the second used only in the shim's verification query. This allows us to instantiate the shim's verification algorithm with a version of $\mathcal{G}$ that does not count towards the total number of queries when bounding the probability of $\mathsf{bad}_{\mathcal{G}}$ in the simulation step.

Related to this, an unfortunate amount of the complexity in the formal proof comes from the need to explicitly count oracle queries when bounding probabilities. In practice, $\mathsf{EasyCrypt}$ in fact requires that oracles that may trigger the event whose probability is being bounded be called a statically bounded number of times. Our formal development uses the manipulations discussed by Barthe et al. [7] to instead bound the adversary. We expect ongoing work on formalizing complexity analysis to better support such analyses in future developments.

Proof Effort and Challenges: An initial version of the formal proof for EDL was developed over the course of 6 person-months by a novice to both cryptography and formal proof. The starting point was Goh and Jarecki's 2003 proof [3], which our initial proof followed and extended with additional failure events. The initial proof for CM was carried out in one person-week by an experienced $\mathsf{EasyCrypt}$ developer, relying heavily on the insights gained during the EDL formalization (in particular, adapting the sequence of games instead of starting from the direct reduction). The development of the common shim, along with adapting the existing proofs to make use of it, took one additional person-week. The effort involved in its ideation is more difficult to quantify, as it occurred continuously through the roughly 6.5 person months of development—and additional reading of related work.

The proof relies on algebraic arguments which form the core of its practical difficulty (past initially formalizing the sequence of games). Further support for symbolic techniques (perhaps inspired by the tools discussed by Barthe et al. [12]) would greatly simplify our proofs, and future proofs in DL-based settings.

Lessons Learned: Our formal efforts started from existing pen-and-paper proofs, based on direct reductions. We first isolated the three steps on paper and almost immediately started formalizing the arguments. Although we converged relatively quickly towards the pattern *embedding, simulation, reduction*, initial formalization efforts which placed a different order on the proof steps were in large part wasted—derivations of group and field arithmetic results were mainly preserved. This—once again—highlights the value of first gaining as full an understanding as possible on paper of the arguments to formalise before starting the machine-checking effort itself.

On the other hand, later iterations to adapt the full proofs to make use of the shims were almost lossless, in the sense that the main changes we needed to make to the proofs were *removals* of repeated arguments. These modifications were only made possible by the identification—through the initial formalizations—of the proofs' common structure.

Striking the right balance between gaining understanding *before* embarking on a formalization effort and gaining understanding *through* the formalization effort remains a delicate exercise. We cannot offer insights, but hope that this simple observation encourages both more thorough pen-and-paper arguments and more deliberate experimentation with formalization.

B. Related Work

We now discuss closely related work, first considering digital signature schemes, and then the state of machine-checked proofs for DL-based cryptographic constructions.

Proofs for digital signature schemes: Tight proofs for digital signatures exist for PSS [13]. EDL, CM and their variants by Katz and Wang [14] and Goh et al. [15] are—to our knowledge—the only DL-based signature schemes equipped with tight proofs. Coron [13], and Bader, Jager, Li and Schäge [16] show the impossibility of obtaining tight reductions in some settings (for FDH, and for signing in multi-user settings). Proofs following our game sequence are not necessarily tight: the final reduction step is where looseness could be introduced, rather than the sequence itself. As such, we believe the techniques presented here are not limited to those schemes for which tight bounds can be established.

Katz and Wang [14] discuss a proof technique—based on claw-free permutations—that applies to PSS as well as to DL-based signing schemes to obtain tight security proofs. Machine-checking their proofs could yield interesting insights in future. Barthe et al [17] formalise a security proof for PSS. Although the proof involves the consideration of faults, it also establishes a tight machine-checked security bound for the security of PSS in the absence of faults, following proofs by Coron and Mandal [13], [18]. The proof's structure is in fact similar to that of the proofs discussed here; considering it as an instance of our shim may help generalize the shim further for broader applicability.

El Kaafarani, Katsumata and Pintore [19] propose a tightly-secure post-quantum signing scheme based on CSIDH [20] and CSI-FiSh [21]. Their proof relies on a generic result on the security of Fiat-Shamir in the Quantum-Random Oracle Model (QROM) due to Kiltz, Lyubashevsky and Schaffner [22]. The proof shape we formalise here does not consider the QROM, and is unlikely to apply in the presence of quantum-capable adversaries. Extensions in this direction would be natural as future work. We note, however, that the formalization of such proof is still on the edge of feasibility [23].

Machine-checked proofs for DL-based cryptography: Although this paper presents the first machine-checked proof for a DL-based signature scheme, other machine-checked proofs exist for DL-based cryptography more generally.

EasyCrypt was used to formalize the security of Cramer-Shoup [24], one-round key exchange protocols [25] and Amazon Web Services' Key Management Service [26].

Only the Cramer-Shoup proofs involve reasoning about arithmetic in cyclic groups similar to that done here, with the others focusing on DL-based key exchange and key encapsulation. (Complexity in these other cases often stems from considering multiple interactive instances with corruption.)

Proofs in other tools, such as F* [27] or CryptoVerif [28] (for Wireguard [29], TLS 1.3 [30], [31], Signal [32], OEKE [33] and HPKE [34]), also focus on protocol reasoning, with little to no reasoning about the group structure involved, and challenges arising from the complexity of the properties being proved rather than inherent complexity in the mathematical arguments involved.

C. Further Generalizations

The security proofs of EDL and CM, as initially presented by Chevallier-Mames [5], are very similar—and indeed carry over very similar objects. In this formalization effort, we purposefully sought out similarities, and attempted to factor them out. We believe the proof pattern extracted in this way could be adapted to other constructions—existing or new—to obtain new proofs of tight security.

In particular, we outline a three-step proof structure, that relies on:

- *embedding* the underlying challenge into random oracle answers;
- *simulating* the zero-knowledge proof; and
- *reducing* from a combination of *soundness* of the zero-knowledge proof and the underlying computational assumption (perhaps via *special soundness*).

We now discuss potential directions for further generalizations.

Embedding in EDL and CM involves *computing* the random oracle answer h from $\mathsf{g_b}$ *and some trapdoor information* d such that h is distributed uniformly at random in $\mathbb{G}$ but such that

$f(\mathsf{d}, \mathsf{h}^{\mathsf{sk}}) = \mathsf{g}_{\mathsf{b}}$. In this context, the embedding operation is in fact a one-time pad, and our proof relies on its malleability. It is worth considering whether the proof technique could be generalised to cases where the secrecy of the embedding is not perfect. This may be required in settings where the embedding function itself (or the extraction function f) is not as simple as in this case.

A related observation is that the relation being proved by the (sound, zero-knowledge, and potentially special sound) proof or argument needs to be tightly integrated with the embedding function: it is only because we can malleably push exponents into the embedding that we can leverage the soundness of the proof system to extract the solution to the given hard problem instance. The occurrence of special soundness in the CM proof may appear—at first glance—more ad hoc, but also highlights a precious ingredient in the proof pattern: for CM-style schemes that commit to the message only late in the signing process, it is important that the witness for the relation be sufficient to solve the target hard problem.

D. Future Work

Beyond the generalizations discussed above, we also note that the results presented here, and associated proof artefacts, also open new directions to push formalization into.

The original EDL scheme was proposed by Chaum and Pedersen [1] for use in a tamper-resistant wallet. Signature computation was meant to be split, with the host producing the value h and a proof of its well-formedness, and the wallet producing the rest of the signature. Direct Anonymous Attestation schemes [35] are very similar in their structure. They are widely deployed in Trusted Platform Modules, and ECC-DAA [36] is now part of a FIDO alliance standard (as ECDAA). This makes DAA schemes excellent targets for formalization on the back of the results presented here.

A more immediate extension would also consider the schemes by Katz and Wang [14] and Goh, Jarecki, Katz and Wang [15], which also benefit from tight discrete logarithm-based reductions. Extending the proof schema—and the corresponding formal shim—to support, notably, Goh, Jarecki, Katz and Wang's techniques to leverage unpredictability—as opposed to full randomness—of the second base h would extend the class of signature schemes whose security could be machine-checked at minimal cost.

Acknowledgments.

This work was supported by Microsoft Research and EPSRC through Microsoft's PhD Scholarship Programme. We thank Constantin Cătălin Drăgan and the anonymous reviewers for useful comments on earlier versions of this paper.

References

[1] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology – CRYPTO'92*, ser. Lecture Notes in Computer Science, E. F. Brickell, Ed., vol. 740. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 16–20, 1993, pp. 89–105.

[2] M. Jakobsson and C. Schnorr, "Efficient oblivious proofs of correct exponentiation," in *Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium*, 1999, pp. 71–86.

[3] E.-J. Goh and S. Jarecki, "A signature scheme as secure as the Diffie-Hellman problem," in *Advances in Cryptology – EUROCRYPT 2003*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 2656. Warsaw, Poland: Springer, Heidelberg, Germany, May 4–8, 2003, pp. 401–415.

[4] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology – EUROCRYPT'96*, ser. Lecture Notes in Computer Science, U. M. Maurer, Ed., vol. 1070. Saragossa, Spain: Springer, Heidelberg, Germany, May 12–16, 1996, pp. 387–398.

[5] B. Chevallier-Mames, "An efficient CDH-based signature scheme with a tight security reduction," in *Advances in Cryptology – CRYPTO 2005*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 14–18, 2005, pp. 511–526.

[6] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, http://eprint.iacr.org/2004/332.

[7] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P.-Y. Strub, *EasyCrypt: A Tutorial*. Cham: Springer International Publishing, 2014, pp. 146–166. [Online]. Available: https://doi.org/10.1007/978-3-319-10082-1_6

[8] G. Barthe, B. Grégoire, and S. Zanella-Béguelin, "Formal certification of code-based cryptographic proofs," *SIGPLAN Not.*, vol. 44, no. 1, p. 90–101, Jan. 2009. [Online]. Available: https://doi.org/10.1145/1594834.1480894

[9] J. B. Almeida, M. Barbosa, G. Barthe, and F. Dupressoir, "Certified computer-aided cryptography: efficient provably secure machine code from high-level implementations," in *ACM CCS 2013: 20th Conference on Computer and Communications Security*, A.-R. Sadeghi, V. D. Gligor, and M. Yung, Eds. Berlin, Germany: ACM Press, Nov. 4–8, 2013, pp. 1217–1230.

[10] ——, "Verifiable side-channel security of cryptographic implementations: Constant-time MEE-CBC," in *Fast Software Encryption – FSE 2016*, ser. Lecture Notes in Computer Science, T. Peyrin, Ed., vol. 9783. Bochum, Germany: Springer, Heidelberg, Germany, Mar. 20–23, 2016, pp. 163–184.

[11] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93: 1st Conference on Computer and Communications Security*, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds. Fairfax, Virginia, USA: ACM Press, Nov. 3–5, 1993, pp. 62–73.

[12] G. Barthe, B. Grégoire, C. Jacomme, S. Kremer, and P.-Y. Strub, "Symbolic methods in computational cryptography proofs," in *CSF 2019: IEEE 32st Computer Security Foundations Symposium*, S. Delaune and L. Jia, Eds. Hoboken, NJ, USA: IEEE Computer Society Press, jun 25-28 2019, pp. 136–151.

[13] J.-S. Coron, "Optimal security proofs for PSS and other signature schemes," in *Advances in Cryptology – EUROCRYPT 2002*, ser. Lecture Notes in Computer Science, L. R. Knudsen, Ed., vol. 2332. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 28 – May 2, 2002, pp. 272–287.

[14] J. Katz and N. Wang, "Efficiency improvements for signature schemes with tight security reductions," in *ACM CCS 2003: 10th Conference on Computer and Communications Security*, S. Jajodia, V. Atluri, and T. Jaeger, Eds. Washington, DC, USA: ACM Press, Oct. 27–30, 2003, pp. 155–164.

[15] E.-J. Goh, S. Jarecki, J. Katz, and N. Wang, "Efficient signature schemes with tight reductions to the Diffie-Hellman problems," *Journal of Cryptology*, vol. 20, no. 4, pp. 493–514, Oct. 2007.

[16] C. Bader, T. Jager, Y. Li, and S. Schäge, "On the impossibility of tight cryptographic reductions," in *Advances in Cryptology – EUROCRYPT 2016, Part II*, ser. Lecture Notes in Computer Science, M. Fischlin and J.-S. Coron, Eds., vol. 9666. Vienna, Austria: Springer, Heidelberg, Germany, May 8–12, 2016, pp. 273–304.

[17] G. Barthe, F. Dupressoir, P.-A. Fouque, B. Grégoire, M. Tibouchi, and J.-C. Zapalowicz, "Making RSA-PSS provably secure against non-random faults," in *Cryptographic Hardware and Embedded Systems – CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds., vol. 8731. Busan, South Korea: Springer, Heidelberg, Germany, Sep. 23–26, 2014, pp. 206–222.

[18] J.-S. Coron and A. Mandal, “PSS is secure against random fault attacks,” in *Advances in Cryptology – ASIACRYPT 2009*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Tokyo, Japan: Springer, Heidelberg, Germany, Dec. 6–10, 2009, pp. 653–666.

[19] A. El Kaafarani, S. Katsumata, and F. Pintore, “Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512,” in *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, ser. Lecture Notes in Computer Science, A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds., vol. 12111. Edinburgh, UK: Springer, Heidelberg, Germany, May 4–7, 2020, pp. 157–186.

[20] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes, “CSIDH: An efficient post-quantum commutative group action,” in *Advances in Cryptology – ASIACRYPT 2018, Part III*, ser. Lecture Notes in Computer Science, T. Peyrin and S. Galbraith, Eds., vol. 11274. Brisbane, Queensland, Australia: Springer, Heidelberg, Germany, Dec. 2–6, 2018, pp. 395–427.

[21] W. Beullens, T. Kleinjung, and F. Vercauteren, “CSI-FiSh: Efficient isogeny based signatures through class group computations,” in *Advances in Cryptology – ASIACRYPT 2019, Part I*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11921. Kobe, Japan: Springer, Heidelberg, Germany, Dec. 8–12, 2019, pp. 227–247.

[22] E. Kiltz, V. Lyubashevsky, and C. Schaffner, “A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model,” in *Advances in Cryptology – EUROCRYPT 2018, Part III*, ser. Lecture Notes in Computer Science, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Tel Aviv, Israel: Springer, Heidelberg, Germany, Apr. 29 – May 3, 2018, pp. 552–586.

[23] D. Unruh, “Post-quantum verification of Fujisaki-Okamoto,” in *Advances in Cryptology – ASIACRYPT 2020, Part I*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds., vol. 12491. Daejeon, South Korea: Springer, Heidelberg, Germany, Dec. 7–11, 2020, pp. 321–352.

[24] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella Béguelin, “Computer-aided security proofs for the working cryptographer,” in *Advances in Cryptology – CRYPTO 2011*, ser. Lecture Notes in Computer Science, P. Rogaway, Ed., vol. 6841. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 14–18, 2011, pp. 71–90.

[25] G. Barthe, J. M. Crespo, Y. Lakhnech, and B. Schmidt, “Mind the gap: Modular machine-checked proofs of one-round key exchange protocols,” in *Advances in Cryptology – EUROCRYPT 2015, Part II*, ser. Lecture Notes in Computer Science, E. Oswald and M. Fischlin, Eds., vol. 9057. Sofia, Bulgaria: Springer, Heidelberg, Germany, Apr. 26–30, 2015, pp. 689–718.

[26] J. B. Almeida, M. Barbosa, G. Barthe, M. Campagna, E. Cohen, B. Grégoire, V. Pereira, B. Portela, P.-Y. Strub, and S. Tasiran, “A machine-checked proof of security for AWS key management service,” in *ACM CCS 2019: 26th Conference on Computer and Communications Security*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM Press, Nov. 11–15, 2019, pp. 63–78.

[27] N. Swamy, C. Hritcu, C. Keller, A. Rastogi, A. Delignat-Lavaud, S. Forest, K. Bhargavan, C. Fournet, P.-Y. Strub, M. Kohlweiss, J.-K. Zinzindohoué, and S. Zanella-Béguelin, “Dependent types and multi-monadic effects in F*,” in *43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. ACM, Jan. 2016, pp. 256–270. [Online]. Available: https://www.fstar-lang.org/papers/mumon/

[28] B. Blanchet, “A computationally sound mechanized prover for security protocols,” *IEEE Trans. Dependable Secur. Comput.*, vol. 5, no. 4, pp. 193–207, 2008. [Online]. Available: https://doi.org/10.1109/TDSC.2007.1005

[29] B. Lipp, B. Blanchet, and K. Bhargavan, “A mechanised cryptographic proof of the wireguard virtual private network protocol,” in *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 231–246. [Online]. Available: https://doi.org/10.1109/EuroSP.2019.00026

[30] K. Bhargavan, B. Blanchet, and N. Kobeissi, “Verified models and reference implementations for the TLS 1.3 standard candidate,” in *2017 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, May 22–26, 2017, pp. 483–502.

[31] B. Blanchet, “Composition theorems for CryptoVerif and application to TLS 1.3,” in *CSF 2018: IEEE 31st Computer Security Foundations Symposium*, S. Chong and S. Delaune, Eds. Oxford, UK: IEEE Computer Society Press, jul 9-12 2018, pp. 16–30.

[32] N. Kobeissi, K. Bhargavan, and B. Blanchet, “Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach,” in *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*. IEEE, 2017, pp. 435–450. [Online]. Available: https://doi.org/10.1109/EuroSP.2017.38

[33] B. Blanchet, “Automatically verified mechanized proof of one-encryption key exchange,” in *CSF 2012: IEEE 25st Computer Security Foundations Symposium*, S. Zdancewic and V. Cortier, Eds. Cambridge, MA, USA: IEEE Computer Society Press, jun 25-27 2012, pp. 325–339.

[34] J. Alwen, B. Blanchet, E. Hauck, E. Kiltz, B. Lipp, and D. Riepel, “Analysing the HPKE standard,” *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1499, 2020. [Online]. Available: https://eprint.iacr.org/2020/1499

[35] E. F. Brickell, J. Camenisch, and L. Chen, “Direct anonymous attestation,” in *ACM CCS 2004: 11th Conference on Computer and Communications Security*, V. Atluri, B. Pfitzmann, and P. McDaniel, Eds. Washington, DC, USA: ACM Press, Oct. 25–29, 2004, pp. 132–145.

[36] L. Chen, D. Page, and N. P. Smart, “On the design and implementation of an efficient daa scheme,” in *Smart Card Research and Advanced Application*, D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 223–237.