An extended abstract of this paper appears in *Advances in Cryptology – EUROCRYPT '01*, Lecture Notes in Computer Science Vol. 2045, B. Pfitzmann ed., Springer-Verlag, 2001. This is the full version.

Identification Protocols Secure Against Reset Attacks

Mihir Bellare* Marc Fischlin† Shafi Goldwasser‡ Silvio Micali§

April 28, 2000

Abstract

We provide identification protocols that are secure even when the adversary can reset the internal state and/or randomization source of the user identifying itself, and when executed in an asynchronous environment like the Internet that gives the adversary concurrent access to instances of the user. These protocols are suitable for use by devices (like smartcards) which when under adversary control may not be able to reliably maintain their internal state between invocations.

Keywords: Identification, zero-knowledge, reset, concurrency, signatures, encryption, authentication.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering.

†Dept. of Mathematics (AG 7.2), Johann Wolfgang Goethe-University, Postfach 111932, 60054 Frankfurt/Main, Germany. E-mail: marc@mi.informatik.uni-frankfurt.de URL: www.mi.informatik.uni-frankfurt.de.

‡MIT Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139, USA. E-Mail: shafi@theory.lcs.mit.edu.

§MIT Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139, USA.

Contents

1 Introduction

An identification protocol enables one entity to identify itself to another as the legitimate owner of some key. This problem has been considered in a variety of settings. Here we are interested in an asymmetric setting. The entity identifying itself is typically called the prover, while the entity to which the prover is identifying itself is called the verifier. The prover holds a secret key sk whose corresponding public key pk is assumed to be held by the verifier.

The adversary's goal is to impersonate the prover, meaning to get the verifier to accept it as the owner of the public key pk. Towards this goal, it is allowed various types of attacks on the prover. In the model of smartcard based identification considered by [18], the adversary may play the role of verifier and interact with the prover, trying to learn something about sk, before making its impersonation attempt. In the model of "Internet" based identification considered by [9, 3, 8], the adversary is allowed to interact concurrently with many different prover "instances" as well as with the verifier. Formal notions of security corresponding to these settings have been provided in the works in question, and there are many protocol solutions for them in the literature.

In this work we consider a novel attack capability for the adversary. We allow it, while interacting with the prover, to reset the prover's internal state. That is, it can "backup" the prover, maintaining the prover's coins, and continue its interaction with the prover. In order to allow the adversary to get the maximum possible benefit from this new capability, we also allow it to have concurrent access to different prover instances. Thus, it can interact with different prover instances and reset each of them at will towards its goal of impersonating the prover. The question of the security of identification protocols under reset attacks was raised by Canetti, Goldreich, Goldwasser and Micali [11], who considered the same issue in the context of zero-knowledge proofs.

1.1 The power of reset attacks

An example. Let us illustrate the power of reset attacks with an example. A popular paradigm for smartcard based identification is to use a proof of knowledge [18]. The prover's public key is an instance of a hard NP language L, and the secret key is a witness to the membership of the public key in L. The protocol enables the prover to prove that it "knows" sk. A protocol that is a proof of knowledge for a hard problem, and also has an appropriate zero-knowledge type property such as being witness hiding [19], is a secure identification protocol in the smartcard model [18].

A simple instance is the zero-knowledge proof of quadratic residuosity of [22]. The prover's public key consists of a composite integer N and a quadratic residue $u \in Z_N^*$. The corresponding secret key is a square root $s \in Z_N^*$ of u. The prover proves that it "knows" a square root of u, as follows. It begins the protocol by picking a random $r \in Z_N^*$ and sending $y = r^2 \bmod N$ to the verifier. The latter responds with a random challenge bit c. The prover replies with $a = rs^c \bmod N$, meaning it returns r if $c = 0$ and $rs \bmod N$ if $c = 1$. The verifier checks that $a^2 \equiv yu^c \bmod N$. (This atomic protocol has an error probability of $1/2$, which can be lowered by sequential repetition. The Fiat-Shamir protocol [20] can be viewed as a parallelized variant of this protocol.)

Now suppose the adversary is able to mount reset attacks on the prover. It can run the prover to get y, feed it challenge 0, and get back $a = r$. Now, it backs the prover up to the step just after it returned y, and feeds it challenge 1 to get answer $a' = rs$. From a and a' it is easily able to extract the prover's secret key s. Thus, this protocol is not secure under reset attacks.

Generalizing from the example, we see that in fact, all proof of knowledge based identification protocols can broken in the same way. Indeed, in a proof of knowledge, the prover is defined to "know a secret" exactly when this secret can be extracted by a polynomial time algorithm (the "extractor") which has oracle access to the prover and is allowed to reset the latter [18, 6]. An attacker allowed a reset attack can simply run the extractor, with the same result, namely it gets the secret. So the bulk

of efficient smartcard based identification protocols in the literature are insecure under reset attacks.

MOUNTING RESET ATTACKS. Resetting or restoring the computational state of a device is particularly simple in the case the device consists of a smartcard which the enemy can capture and experiment with. If the card is manufactured with secure hardware, the enemy may not be able to read its secret content, but it could disconnect its battery so as to restore the card's secret internal content to some initial state, and then re-insert the battery and use it with that state a number of times. If the smart card implements a proof of knowledge prover for ID purposes, then such an active enemy may impersonate the prover later on.

Other scenarios in which such an attack can be realized is if an enemy is able to force a crash on the device executing the prover algorithm, in order to force it to resume computation after the crash in an older "computational state", thereby forcing it to essentially reset itself.

CAN WE USE RESETTABLE ZERO-KNOWLEDGE? Zero-knowledge proofs of membership secure under reset attack do exist [11], but for reasons similar to those illustrated above, are not proofs of knowledge. Accordingly, they cannot be used for identification under a proof of knowledge paradigm. One of the solution paradigms we illustrate later however will show how proofs of membership, rather than proofs of knowledge, can be used for identification.

1.2 Notions of security

Towards the goal of proving identification protocols secure against reset attacks, we first discuss the notions of security we define and use.

We distinguish between two types of resettable attacks CR1 (Concurrent-Reset-1) and CR2 (Concurrent-Reset-2). In a CR1 attack, Vicky (the adversary) may interact concurrently, in the role of verifier, with many instances of the prover Alice, resetting Alice to initial conditions and interleaving executions, hoping to learn enough to be able to impersonate Alice in a future time. Later, Vicky will try to impersonate Alice, trying to identify herself as Alice to Bob (the verifier).

In a CR2 attack, Vicky, *while* trying to impersonate Alice (i.e attempting to identify herself as Alice to Bob the verifier), may interact concurrently, in the role of verifier, with many instances of the prover Alice, resetting Alice to initial conditions and interleaving executions. Clearly, a CR1 attack is a special case of a CR2 attack.

A definition of what it means for Vicky to win in the CR1 setting is straightforward: Vicky wins if she can make the verifier Bob accept. In the CR2 setting Vicky can make the verifier accept by simply being the woman-in-the-middle, passing messages back and forth between Bob and Alice. The definitional issues are now much more complex because the woman-in-the-middle "attack" is not really an attack and the definition must take this into account. We address these issues based on definitional ideas from [9, 8], specifically by assigning session-ids to each completed execution of an ID protocol, which the prover must generate and the verifier accept at the completion of the execution. For reasons of brevity we do not discuss the CR2 setting much in this abstract, and refer the reader to the full version of this paper [5].

We clarify that the novel feature of our work is the consideration of reset attacks for identification. However our settings are defined in such a way that the traditional concurrent attacks as considered by [9, 17] and others are incorporated, so that security against these attacks is achieved by our protocols.

1.3 Four paradigms for identification secure against reset attack

As we explained above, the standard proof of knowledge based paradigm fails to provide identification in the resettable setting. In that light, it may not be clear how to even prove the existence of a solution to the problem. Perhaps surprisingly however, not only can the existence of solutions be proven under the minimal assumption of a one-way function, but even simple and efficient solutions can be designed.

This is done in part by returning to some earlier paradigms. Zero-knowledge proofs of knowledge and identification are so strongly linked in contemporary cryptography that it is sometimes forgotten that these in fact replaced earlier identification techniques largely due to the efficiency gains they brought. In considering a new adversarial setting it is thus natural to first return to older paradigms and see whether they can be "lifted" to the resettable setting. We propose in particular signature and encryption based solutions for resettable identification and prove them secure in both the CR1 and the CR2 settings. We then present a general method for transforming identification protocols secure in a concurrent but non-reset setting to ones secure in a reset setting. Finally we return to the zero-knowledge ideas and provide a new paradigm based on zero-knowledge proofs of membership as opposed to proofs of knowledge.

SIGNATURE BASED IDENTIFICATION. The basic idea of the signature based paradigm is for Alice convinces Bob that she is Alice, by being "able to" sign random documents of Bob's choice. This is known (folklore) to yield a secure identification scheme in the serial non-reset setting of [18] as long as the signature scheme is secure in the sense of [23]. It is also known to be secure in the concurrent non-reset setting [3]. But it fails in general to be secure in the resettable setting because an adversary can obtain signatures of different messages under the same prover coins. What we show is that the paradigm yields secure solutions in the resettable setting if certain special kinds of signature schemes are used. (The signing algorithm should be deterministic and stateless.) In the CR1 setting the basic protocol using such signature schemes suffices. The CR2 setting is more complex and we need to modify the protocol to include "challenges" sent by the prover. Since signature schemes with the desired properties exist (and even efficient ones exist) we obtain resettable identification schemes proven secure under minimal assumptions for both the CR1 and the CR2 settings, and also obtain some efficient specific protocols.

ENCRYPTION BASED IDENTIFICATION. In the encryption based paradigm, Alice convinces Bob she is Alice, by being "able to" decrypt ciphertexts which Bob created. While the basic idea goes back to symmetric authentication techniques of the seventies, modern treatments of this paradigm appeared more recently in [15, 3, 17] but did not consider reset attacks. We show that under an appropriate condition on the encryption scheme —namely that it be secure against chosen-ciphertext attacks— a resettable identification protocol can be obtained. As before the simple solution for the CR1 setting needs to be modified before it will work in the CR2 setting.

TRANSFORMING STANDARD PROTOCOLS. Although Fiat-Shamir like identification protocols are not secure in the context of reset attacks, with our third paradigm we show how to turn practical identification schemes into secure ones in the CR1 and CR2 settings. The solution relies on the techniques introduced in [11] and utilizes pseudorandom functions and trapdoor commitments. It applies to most of the popular identification schemes, like Fiat-Shamir [20], Okamoto-Schnorr [31, 28] or Okamoto-Guillou-Quisquater [25, 28].

ZK PROOF OF MEMBERSHIP BASED IDENTIFICATION. In the zero-knowledge proofs of membership paradigm, Alice convinces Bob she is Alice, by being "able to" prove membership in a hard language L, rather than by proving she has a witness for language L. She does so by employing a resettable zero-knowledge proof of language membership for L as defined in [11] . Both Alice and Bob will need to have a public-key to enable the protocol. Alice's public-key defines who she is, and Bob's public-key enables him to verify her identity in a secure way. We adopt the general protocol for membership in NP languages of [11] for the purpose of identification. The identification protocols are constant round. What makes this work is the fact that the protocol for language membership ($x \in L$) being zero-knowledge implies "learning nothing" about x in a very strong sense — a verifier cannot subsequently convince anyone else that $x \in L$ with non-negligible probability. We note that while we can make this approach work using resettable zero-knowledge proofs, it does not seem to work using resettable

witness indistinguishable proofs for ID protocols.

PERSPECTIVE. Various parts of the literature have motivated the study of zero-knowledge protocols secure against strong attacks such as concurrent or reset in part by the perceived need for such tools for the purpose of applications such as identification in similar attack settings. While the tools might be sufficient for identification, they are not necessary. Our results demonstrate that identification is much easier than zero-knowledge and the latter is usually an overkill for the former.

2 Definitions

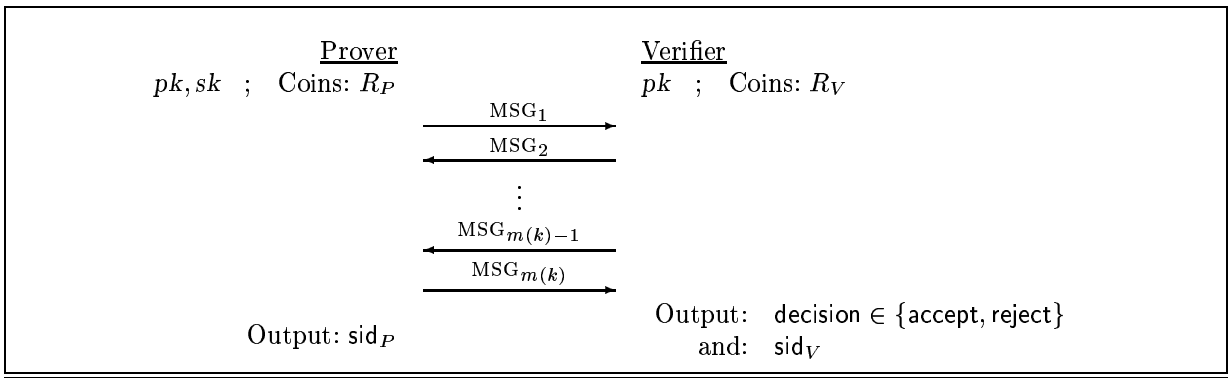
The adversary model here, allowing reset attacks in a concurrent execution setting, is the strongest one for identification considered to date. It is convenient to define two versions of the model: Concurrent-Reset-1 (CR1) and Concurrent-Reset-2 (CR2). While both models allow concurrent reset attacks on provers, in CR1 —which models smartcard based identification and extends the setting of [18]— the adversary is allowed access to provers only prior to its attempt to convince the verifier to accept, while in CR2 —which models network or "Internet" based identification and extends the setting of [9]— the adversary maintains access to the provers even while trying to convince the verifier to accept. The split enables us to take an incremental approach both to the definitions and to the design of protocols, considering first the simpler CR1 setting and then showing how to lift the ideas to the more complex CR2 setting. In this section we present definitions for the CR1 case obtained by adapting and extending [18], and definitions for the CR2 case based on ideas of [9, 8].

NOTATION.If $A(\cdot,\cdot,\ldots)$ is a randomized algorithm then $y \leftarrow A(x_1, x_2, \ldots; R)$ means y is assigned the unique output of the algorithm on inputs $x_1, x_2, \ldots$ and coins R, while $y \leftarrow A(x_1, x_2, \ldots)$ is shorthand for first picking R at random (from the set of all strings of some appropriate length) and then setting $y \leftarrow A(x_1, x_2, \ldots; R)$. If $x_1, x_2, \ldots$ are strings then $x_1 \| x_2 \| \cdots$ denotes an encoding under which the constituent strings are uniquely recoverable. It is assumed any string x can be uniquely parsed as an encoding of some sequence of strings. The empty string is denoted ε.

SYNTAX OF IDENTIFICATION PROTOCOLS.An identification protocol proceeds as depicted in Figure 1. The prover has a secret key sk whose matching public key pk is held by the verifier. (In practice the prover might provide its public key, and the certificate of this public key, as part of the protocol, but this is better slipped under the rug in the model.) Each party computes its next message as a function of its keys, coins and the current conversation prefix. The number of moves $m(k)$ is odd so that the first and last moves belong to the prover. (An identification protocol is initiated by the prover who at the very least must provide a request to be identified.) At the end of the protocol the verifier outputs a decision to either accept or reject. Each party may also output a session id. (Sessions ids are relevant in the CR2 setting but can be ignored for the CR1 setting.) A particular protocol is described by a (single) *protocol description* function $\mathcal{ID}$ which specifies how all associated processes —key generation, message computation, session id or decision computation— are implemented. (We say that $\mathcal{ID}$ is for the CR1 setting if $\mathsf{sid}_P = \mathsf{sid}_V = \varepsilon$, meaning no session ids are generated.) The second part of Figure 1 shows how it works: the first argument to $\mathcal{ID}$ is a keyword —one of keygen, prvmsg, vfmsg, prvsid, vfend— which invokes the subroutine responsible for that function on the other arguments.

COMPLETENESS.Naturally, a correct execution of the protocol (meaning one in the absence of an adversary) should lead the verifier to accept. To formalize this "completeness" requirement we consider an *adversary-free execution* of the protocol $\mathcal{ID}$ which proceeds as described in the following experiment:

$(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$; Choose tapes R_P, R_V at random
$\text{MSG}_1 \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \varepsilon; R_P)$
`For` $j = 1$ `to` $\lfloor m(k)/2 \rfloor$ `do`
  $\text{MSG}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}; R_V)$



$(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$ — Randomized process to generate a public key pk and matching secret key sk
$\text{MSG}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \text{MSG}_1 \| \cdots \text{MSG}_{2j}; R_P)$ — $(1 \leq 2j+1 \leq m(k))$ Next prover message as a function of secret key, conversation prefix and coins R_P
$\text{MSG}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}; R_V)$ — $(2 \leq 2j \leq m(k)-1)$ Next verifier message as a function of public key, conversation prefix and coins R_V
$\mathsf{sid}_P \leftarrow \mathcal{ID}(\mathsf{prvsid}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{m(k)}; R_P)$ — Prover's session id as a function of secret key, full conversation and coins
$\mathsf{sid}_V \| \mathsf{decision} \leftarrow \mathcal{ID}(\mathsf{vfend}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{m(k)}; R_V)$ — Verifier session id and decision (accept or reject) as a function of public key, full conversation and coins

Figure 1: The prover sends the first and last messages in an $m(k)$-move identification protocol at the end of which the verifier outputs a decision and each party optionally outputs a session id. The *protocol description* function $\mathcal{ID}$ specifies all processes associated to the protocol.

$\text{MSG}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_P)$
`EndFor`
$\mathsf{sid}_P \leftarrow \mathcal{ID}(\mathsf{prvsid}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{m(k)}; R_P)$
$\mathsf{sid}_V \| \mathsf{decision} \leftarrow \mathcal{ID}(\mathsf{vfend}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{m(k)}; R_V)$

The *completeness condition* is that, in the above experiment, the probability that $\mathsf{sid}_P = \mathsf{sid}_V$ and $\mathsf{decision} = \mathsf{accept}$ is 1. (The probability is over the coin tosses of $\mathcal{ID}(\mathsf{keygen}, k)$ and the random choices of R_P, R_V.) As always, the requirement can be relaxed to only ask for a probability close to one.

EXPERIMENTS AND SETTINGS.Fix an identification protocol description function $\mathcal{ID}$ and an adversary I. Associated to them is $\mathbf{Experiment}^{\text{id-cr1}}_{\mathcal{ID},I}(k)$, depicted in Figure 2, which is used to define the security of $\mathcal{ID}$ in the CR1 setting. (In this context it is understood that $\mathcal{ID}$ is for the CR1 setting, meaning does not produce session ids.) $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$, depicted in Figure 3, is used to define the security of $\mathcal{ID}$ in the CR2 setting.The experiment gives the adversary appropriate access to prover instance oracles $\mathsf{Prover}^1, \mathsf{Prover}^2, \ldots$ and a single verifier oracle, let it query these subject to certain restrictions imposed by the experiment, and then determine whether it "wins". The interface to the prover instance oracles and the verifier oracle (which, in the experiment, are implicit, never appearing by name) is via oracle queries; the experiment enumerates the types of queries and shows how answers are provided to them.

Eachexperiment begins with some initializations which include choosing of the keys. Then the adversary is invoked on input the public key. A WakeNewProver query activates a new prover instance Prover^p by picking a random tape R_p for it. (A random tape for a prover instance is chosen exactly once and all messages of this prover instance are then computed with respect to this tape. The

Experiment $\mathbf{Experiment}^{\text{id-cr1}}_{\mathcal{ID},I}(k)$ — Execution of protocol $\mathcal{ID}$ with adversary I and security parameter k in the CR1 setting

Initialization:
(1) $(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$ // Pick keys via randomized key generation algorithm //
(2) Choose tape R_V for verifier at random ; $C_V \leftarrow 0$ // Coins and message counter for verifier //
(3) $p \leftarrow 0$ // Number of active prover instances //

Execute adversary I on input pk and reply to its oracle queries as follows:

- When I makes query WakeNewProver // Activate a new prover instance //
  (1) $p \leftarrow p + 1$; Pick a tape R_p at random ; `Return` p
- When I makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j})$ with $0 \le 2j < m(k)$ and $1 \le i \le p$
  (1) `If` $C_V \neq 0$ `then Return` $\perp$ // Interaction with prover instance allowed only before interaction with verifier begins //
  (2) $\text{MSG}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_i)$
  (3) `Return` MSG_{2j+1}
- When I makes query $\mathsf{Send}(\mathsf{vfmsg}, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1})$ with $1 \le 2j - 1 \le m(k)$
  (1) $C_V \leftarrow C_V + 2$
  (2) `If` $2j < C_V$ `then Return` $\perp$ // Not allowed to reset the verifier //
  (3) `If` $2j - 1 < m(k) - 1$ `then` $\text{MSG}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}; R_V)$; `Return` MSG_{2j}
  (4) `If` $2j - 1 = m(k)$ `then` $\mathsf{decision} \leftarrow \mathcal{ID}(\mathsf{vfend}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_V)$
  (5) `Return` decision

Did I win? When I has terminated set $\text{WIN}_I = \mathsf{true}$ if $\mathsf{decision} = \mathsf{accept}$.

Figure 2: Experiment describing execution of identification protocol $\mathcal{ID}$ with adversary I and security parameter k in the CR1 setting.

tape of a specific prover instance cannot be changed, or "reset", once chosen.) A $\mathsf{Send}(\mathsf{prvmsg}, i, x)$ query —viewed as sent to prover instance Prover^i— results in the adversary being returned the next prover message computed as $\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_i)$. (It is assumed that $x = \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}$ is a *valid conversation prefix*, meaning contains an even number of messages $2j < m(k)$, else the query is not valid.) Resetting is captured by allowing arbitrary (valid) conversation prefixes to be queried. (For example the adversary might try $\text{MSG}_1 \| \text{MSG}_2$ for many different values of MSG_2, corresponding to successively resetting the prover instance to the point where it receives the second protocol move.) Concurrency is captured by the fact that any activated prover instances can be queried.

A $\mathsf{Send}(\mathsf{vfmsg}, x)$ query is used to invoke the verifier on a conversation prefix x and results in the adversary being returned either the next verifier message computed as $\mathcal{ID}(\mathsf{vfmsg}, pk, x; R_V)$ — this when the verifier still has a move to make— or the decision computed as $\mathcal{ID}(\mathsf{vfend}, pk, x; R_V)$ —this when x corresponds to a full conversation. (Here R_V was chosen at random in the experiment initialization step. It is assumed that $x = \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}$ is a valid conversation prefix, meaning contains an odd number of messages $1 \le 2j - 1 \le m(k)$, else the query is not valid.) Unlike a prover instance, resetting the (single) verifier instance is not allowed. (Our signature and encryption based protocols are actually secure even if verifier resets are allowed, but since the practical need to consider this attack is not apparent, the definition excludes it.) This is enforced explicitly in the experiments via the verifier message counter C_V.

We now come to the difference in the two settings:

Experiment$^{\text{id-cr}}_{\mathcal{ID},I}(k)$ — Execution of protocol $\mathcal{ID}$ with adversary I and security parameter k in the CR2 setting

Initialization:
(1) $(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$ // Pick keys via randomized key generation algorithm //
(2) Choose tape R_V for verifier at random ; $C_V \leftarrow 0$ // Coins and message counter for verifier //
(3) $p \leftarrow 0$ // Number of active prover instances //

Execute adversary I on input pk and reply to its oracle queries as follows:

- When I makes query WakeNewProver // Activate a new prover instance //
  (1) $p \leftarrow p+1$; $\mathsf{SID}_p \leftarrow \emptyset$; Pick a tape R_p at random ; Return p
- When I makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j})$ with $0 \le 2j < m(k)$ and $1 \le i \le p$
  (1) $\text{MSG}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_i)$; $s \leftarrow \text{MSG}_{2j+1}$
  (2) If $2j+1 = m(k)$ then
    – $\mathsf{sid} \leftarrow \mathcal{ID}(\mathsf{prvsid}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j+1}; R_i)$; $s \leftarrow s \| \mathsf{sid}$
    – $\mathsf{SID}_i \leftarrow \mathsf{SID}_i \cup \{\mathsf{sid}\}$
  (3) Return s
- When I makes query $\mathsf{Send}(\mathsf{vfmsg}, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1})$ with $1 \le 2j-1 \le m(k)$
  (1) $C_V \leftarrow C_V + 2$
  (2) If $2j < C_V$ then Return $\perp$ // Not allowed to reset the verifier //
  (3) If $2j-1 < m(k)-1$ then $\text{MSG}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}; R_V)$; Return MSG_{2j}
  (4) If $2j-1 = m(k)$ then $\mathsf{sid}_V \| \mathsf{decision} \leftarrow \mathcal{ID}(\mathsf{vfend}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_V)$;
  Return $\mathsf{sid}_V \| \mathsf{decision}$

Did I win? When I has terminated set $\text{WIN}_I = \mathsf{true}$ if either of the following are true:
(1) $\mathsf{decision} = \mathsf{accept}$ and $\mathsf{sid}_V \notin \mathsf{SID}_1 \cup \cdots \cup \mathsf{SID}_p$.
(2) There exist $1 \le a < b \le p$ with $\mathsf{SID}_a \cap \mathsf{SID}_b \neq \emptyset$

Figure 3: Experiment describing execution of identification protocol $\mathcal{ID}$ with adversary I and security parameter k in the CR2 setting.

CR1 setting: The adversary's actions are divided into two phases. In the first phase it interacts with the prover instances, not being allowed to interact with the verifier; in the second phase it is denied access to the prover instances and tries to convince the verifier to accept. **Experiment**$^{\text{id-cr1}}_{\mathcal{ID},I}(k)$ enforces this by returning $\perp$ in reply to a $\mathsf{Send}(\mathsf{prvmsg}, i, x)$ unless $C_V = 0$.

CR2 setting: The prover instances and the verifier instance are available simultaneously to the adversary. In particular it can relay message back and forth between them.

WHAT'S A WIN? In the CR1 setting it is easy to say what it should mean for the adversary to "win:" it should make the verifier instance accept. The parameter WIN_I is set accordingly in **Experiment**$^{\text{id-cr1}}_{\mathcal{ID},I}(k)$. What it means for the adversary to "win" is less clear in the CR2 setting because here there is one easy way for the adversary to make the verifier accept: play "man in the middle" between the verifier and some prover instance, relaying messages back and forth between them until the verifier accepts. Yet, it is clear that this is not really an attack; there is no harm in the verifier accepting under these conditions since in fact it was actually talking to the prover. Rather this example highlights the fact that the definitional issues of the second setting are significantly more challenging than those of the first setting: how exactly do we say what it means for the adversary

to win? Luckily, however, this problem has already been solved. The first proposed definition, due to Bellare and Rogaway [9], is based on the idea of "matching conversations" and corresponds to a very stringent security requirement. Another possible definition is that of [8] which uses the idea of "matching session ids." (The idea goes back to Bellare, Petrank, Rackoff and Rogaway, 1996.) We will use the latter definitional approach.

View a session id shared between a prover instance and the verifier as a "connection name," enabling the verifier to differentiate between different prover instances. It is not secret, and in particular will be given to the adversary. (In setting one, even though there are many prover instances, a session id is not necessary to differentiate them from the point of view of the verifier because only one prover instance can interact with the verifier at any time.) In the absence of an adversary, the session ids output by a prover instance and the verifier at the end of their interaction must be the same, but with high probability no two different prover instances should have the same session id, since otherwise the verifier cannot tell them apart. Victory for the adversary now will correspond to making the verifier accept with a session id not held by any prover instance. (We also declare the adversary victorious if it "confuses" the verifier by managing to make two different prover instances output the same session id.) The parameter WIN_I is set accordingly in $\textbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$. Session ids are public in the sense that the adversary gets to see those created by any instances with which it interacts.

DEFINITION OF SECURITY. The experiments indicate under what conditions adversaries are declared to "win." The definition of the protocol is responsible for ensuring that both parties reject a received conversation prefix if it is inconsistent with their coins. It is also assumed that the adversary never repeats an oracle query. We can now provide definitions of security for protocol $\mathcal{ID}$.

Definition 2.1 [Security of an ID protocol in the CR1 setting] Let $\mathcal{ID}$ be an identification protocol description for the CR1 setting. Let I be an adversary (called an *impersonator* in this context) and let k be the security parameter. The *advantage* of impersonator I is

$$\textbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) = \Pr[\,\text{WIN}_I = \mathsf{true}\,]$$

where the probability is with respect to $\textbf{Experiment}^{\text{id-cr1}}_{\mathcal{ID},I}(k)$. Protocol $\mathcal{ID}$ is said to be *polynomially-secure in the* CR1 *setting* if $\textbf{Adv}^{\text{id-cr1}}_{\mathcal{ID}}(\cdot)$ is negligible for any impersonator I of time-complexity polynomial in k. ▌

We adopt the convention that the *time-complexity* $t(k)$ of an adversary I is the execution time of the entire experiment $\textbf{Experiment}^{\text{id-cr1}}_{\mathcal{ID},I}(k)$, including the time taken for initialization, computation of replies to adversary oracle queries, and computation of WIN_I. We also define the *query-complexity* $q(k)$ of I as the number of $\mathsf{Send}(\mathsf{prvmsg},\cdot,\cdot)$ queries made by I in $\textbf{Experiment}^{\text{id-cr1}}_{\mathcal{ID},I}(k)$. It is always the case that $q(k) \leq t(k)$ so an adversary of polynomial time-complexity has polynomial query-complexity. These definitions and conventions can be ignored if polynomial-security is the only concern, but simplify concrete security considerations to which we will pay some attention later.

A definition of security for the CR2 setting can be found in [5].

Definition 2.2 [Security of an ID protocol in the CR2 setting] Let $\mathcal{ID}$ be an identification protocol description. Let I be an adversary (called an *impersonator* in this context) and let k be the security parameter. The *advantage* of impersonator I is

$$\textbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) = \Pr[\,\text{WIN}_I = \mathsf{true}\,]$$

where the probability is with respect to $\textbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$. Protocol $\mathcal{ID}$ is said to be *polynomially-secure in the* CR2 *setting* if $\textbf{Adv}^{\text{id-cr2}}_{\mathcal{ID}}(\cdot)$ is negligible for any impersonator I of time-complexity polynomial in k. ▌

We adopt the same conventions regarding time and query complexity as above.

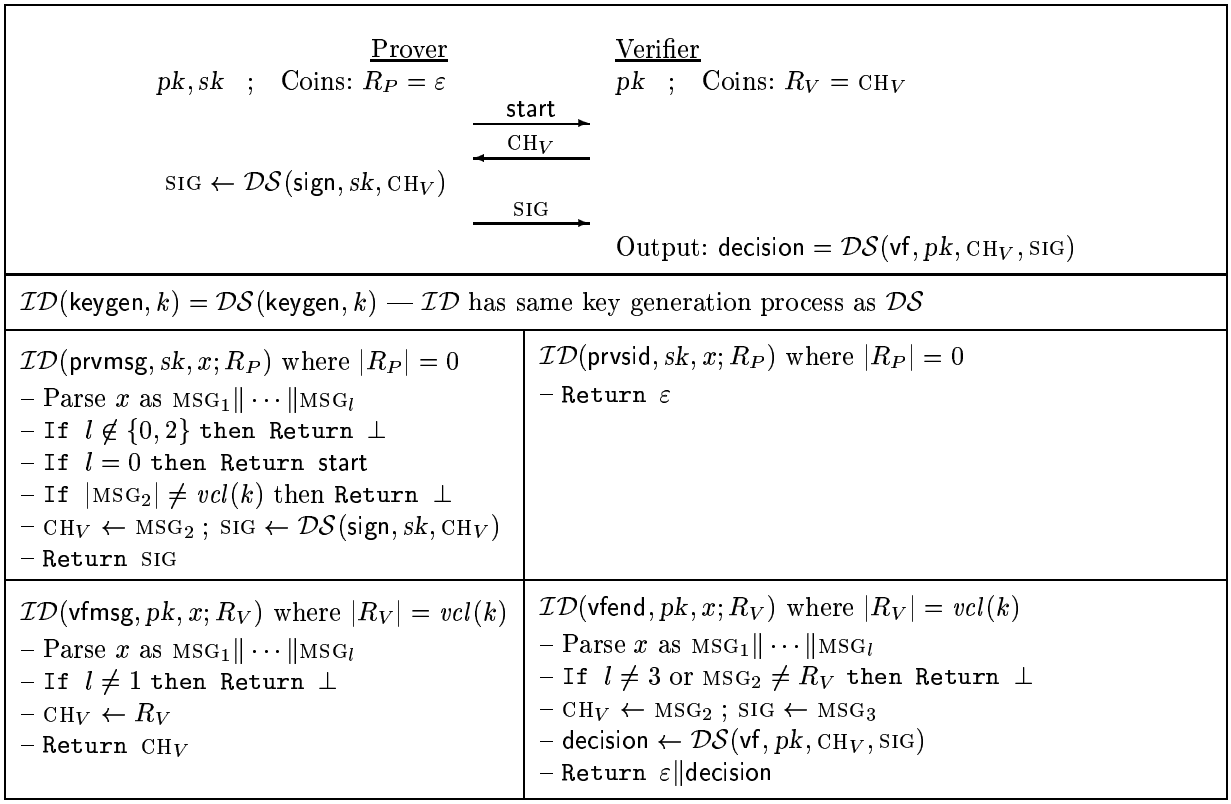


Figure 4: Reset-secure identification protocol $\mathcal{ID}$ for the CR1 setting based on a deterministic, stateless digital signature scheme $\mathcal{DS}$: Schematic followed by full protocol description.

More. Appendix A contains more information about the notions including comparison with previous definitions in the literature.

3 CR1-secure Identification protocols

Four paradigms are illustrated: signature based, encryption based, identification based, and zero-knowledge based.

3.1 A signature based protocol

We assume knowledge of background in digital signatures as summarized in Appendix B.1.

Signature based identification. A natural identification protocol is for the verifier to issue a random challenge CH_V and the prover respond with a signature of CH_V computed under its secret key sk. (Prefix the protocol with an initial start move by the prover to request start of an identification process, and you have a three move protocol.) This simple protocol can be proven secure in the serial, non-resettable (ie. standard smartcard) setting of [18] as long as the signature scheme meets the notion of security of [23] provided in Definition B.1. (This result seems to be folklore.) The same protocol has also been proven to provide authentication in the concurrent, non-resettable (ie. standard network) setting [3]. (The intuition in both cases is that the only thing an adversary can do with a prover oracle is feed it challenge strings and obtain their signatures, and if the scheme is secure against chosen-message attack this will not help the adversary forge a signature of a challenge issued by the verifier unless it guesses the latter, and the probability of the last event can be made small by using

a long enough challenge.) This protocol is thus a natural candidate for identification in the resettable setting.

However this protocol does not always provide security in the resettable setting. The intuition described above breaks down because resetting allows an adversary to obtain the signatures of different messages under the same set of coins. (It can activate a prover instance and then query it repeatedly with different challenges, thereby obtaining their signatures with respect to a fixed set of coin tosses.) As explained in Appendix B.1, this is not covered by the usual notion of a chosen-message attack used to define security of signature schemes in [23]. And indeed, for many signature schemes it is possible to forge the signature of a new message if one is able to obtain the signatures of several messages under one set of coins. Similarly, if the signing algorithm is stateful, resetting allows an adversary to make the prover release several signatures computed using one value of the state variable —effectively, the prover does not get a chance to update its state is it expects to— again leading to the possibility of forgery on a scheme secure in the standard sense.

The solution is simple: restrict the signature scheme to be stateless and deterministic. In Appendix B.1we explain how signatures schemes can be imbued with these attributes so that stateless, deterministic signature schemes are available.

PROTOCOL AND SECURITY. Let $\mathcal{DS}$ be a deterministic, stateless signature scheme. Figure 4 illustrates the flows of the associated identification protocol $\mathcal{ID}$and then provides the protocol description. (The latter includes several checks omitted in the picture but important for security against resets.)A parameter of the protocol is the length $vcl(k)$ of the verifier's random challenge. The prover is deterministic and has random tape ε while the verifier's random tape is CH_V. Refer to Definition 2.1 and Definition B.1 for the meanings of terms used in the theorem below, and to Section D.1 for the proof.

Theorem 3.1 [Concrete security of the signature based ID scheme in the CR1 setting] Let $\mathcal{DS}$ be a deterministic, stateless signature scheme, let $vcl(\cdot)$ be a polynomially-bounded function, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 4. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR1 setting then there exists a forger F attacking $\mathcal{DS}$ such that

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{ds}}_{\mathcal{DS},F}(k) + \frac{q(k)}{2^{vcl(k)}} \,. \tag{1}$$

Furthermore F has time-complexity $t(k)$ and makes at most $q(k)$ signing queries in its chosen-message attack on $\mathcal{DS}$. ▌

This immediately implies the following:

Corollary 3.2 [Polynomial-security of the signature based ID scheme in the CR1 setting] Let $\mathcal{DS}$ be a deterministic, stateless signature scheme, let $vcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 4. If $\mathcal{DS}$ is polynomially-secure then $\mathcal{ID}$ is polynomially-secure in the CR1 setting. ▌

Corollary 3.2 together with Proposition B.2 imply:

Corollary 3.3 [Existence of an ID scheme polynomially-secure in the CR1 setting] Assume there exists a one-way function. Then there exists an identification scheme that is polynomially-secure in the CR1 setting.

This means that we can prove the existence of an identification protocol secure in the CR1 setting under the minimal complexity assumption of a one-way function.

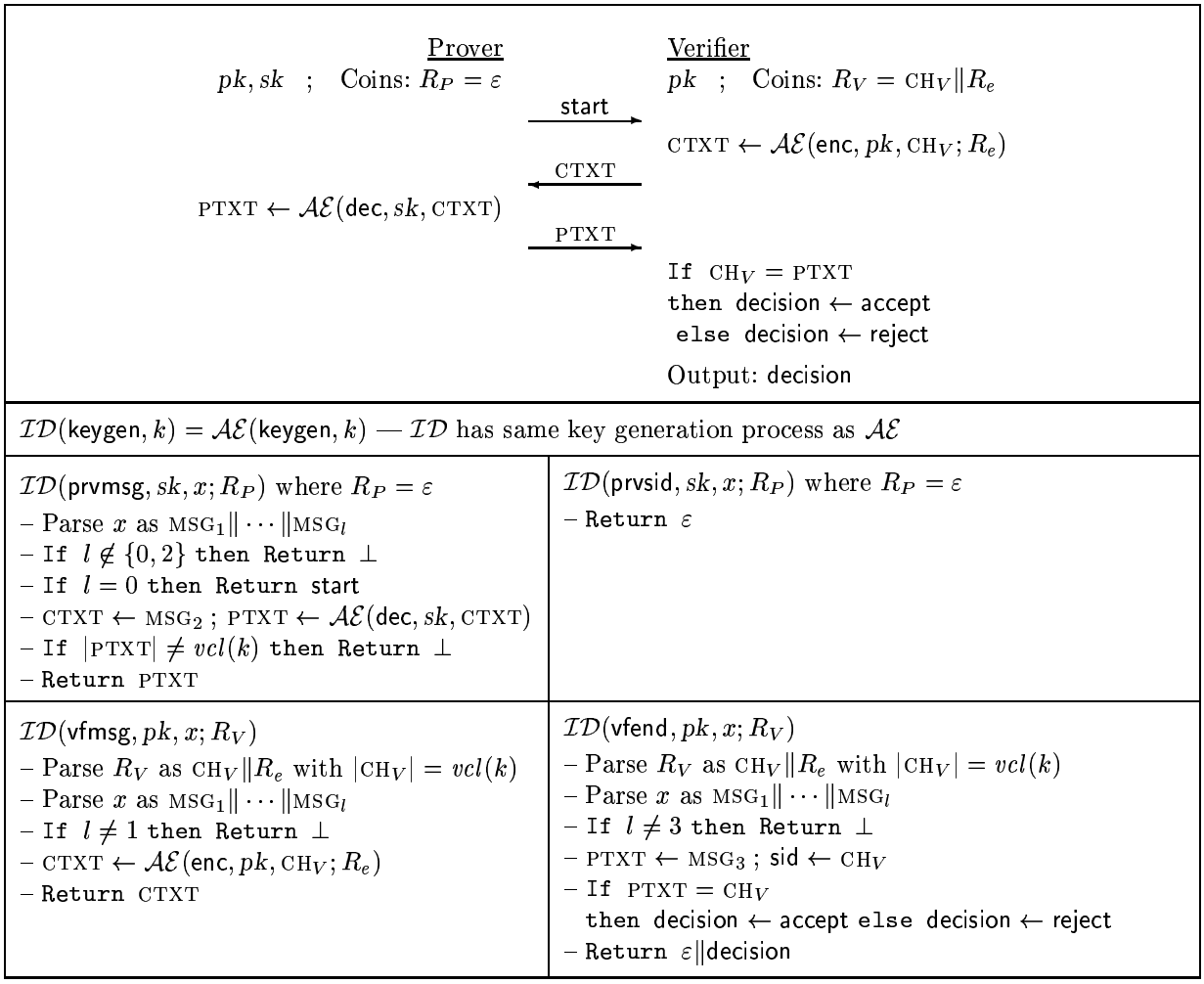


Figure 5: Reset-secure identification protocol $\mathcal{ID}$ for the CR1 setting based on a chosen-ciphertext attack secure asymmetric encryption scheme $\mathcal{AE}$: Schematic followed by full protocol description.

3.2 An encryption based protocol

ENCRYPTION BASED IDENTIFICATION. The idea is simple: the prover proves its identity by proving its ability to decrypt a ciphertext sent by the verifier. This basic idea goes back to early work in entity authentication where the encryption was usually symmetric (ie. private-key based). These early protocols however had no supporting definitions or analysis. The first "modern" treatment is that of [15] who considered the paradigm with regard to providing deniable authentication and identified non-malleability under chosen-ciphertext attack —equivalently, indistinguishability under chosen-ciphertext attack [4, 15]— as the security property required of the encryption scheme. Results of [3, 17, 15] imply that the protocol is a secure identification scheme in the concurrent non-reset setting, but reset attacks have not been considered before.

PROTOCOL AND SECURITY. Let $\mathcal{AE}$ be an asymmetric encryption scheme polynomially-secure against chosen-ciphertext attack. Figure 5 illustrates the flows of the associated identification protocol $\mathcal{ID}$ and then provides the protocol description. A parameter of this protocol is the length $vcl(k)$ of the verifier's random challenge. The verifier sends the prover a ciphertext formed by encrypting a random challenge, and the prover identifies itself by correctly decrypting this to send the verifier back the challenge. The prover is deterministic, having random tape ε. We make the coins R_e used by the

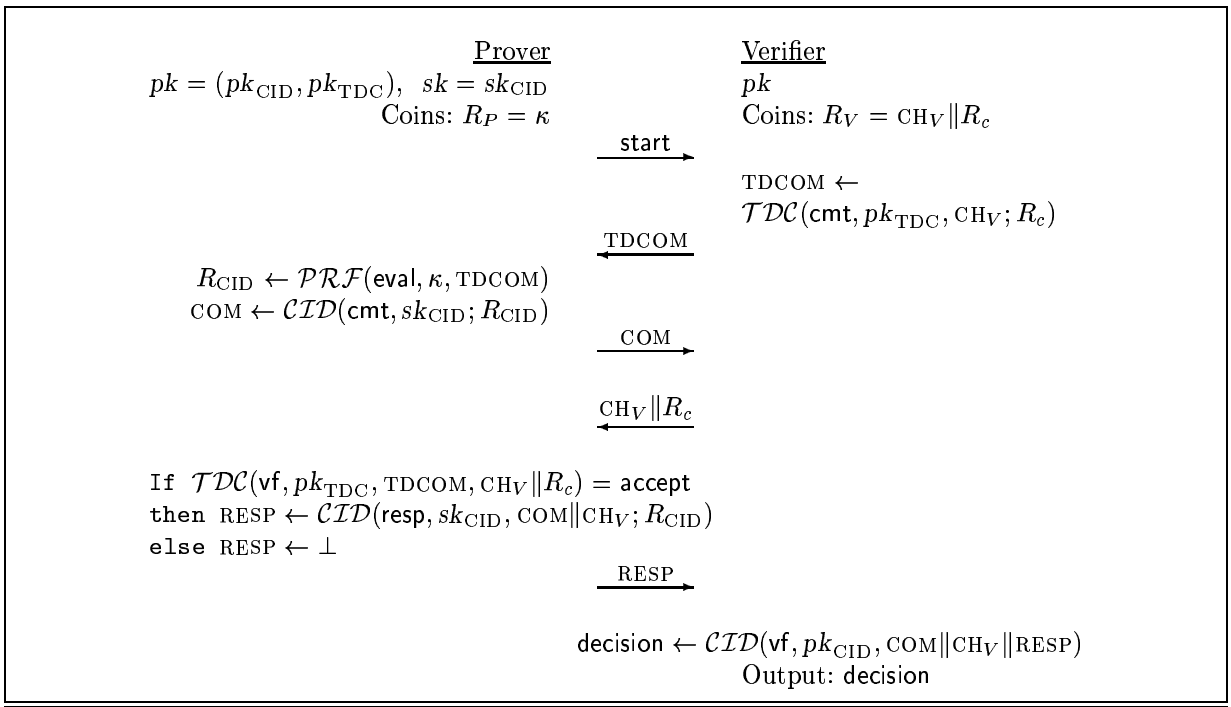


Figure 6: Reset-secure identification protocol $\mathcal{ID}$ for the CR1 setting based on an identification scheme $\mathcal{CID}$ secure against non-resetting CR1 attacks

encryption algorithm explicit, so that the verifier's random tape consists of the challenge —a random string of length $vcl(k)$ where vcl is a parameter of the protocol— and coins sufficient for one invocation of the encryption algorithm. Refer to Definition 2.1 and Definition B.3 for the meanings of terms used in the theorem below, and to Section D.2 for the proof.

Theorem 3.4 [Concrete security of the encryption based ID scheme in the CR1 setting] Let $\mathcal{AE}$ be an asymmetric encryption scheme, let $vcl(\cdot)$ a polynomially-bounded function, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 5. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR1 setting then there exists an eavesdropper E attacking $\mathcal{AE}$ such that

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + \frac{2q(k)+2}{2^{vcl(k)}} \,. \tag{2}$$

Furthermore E has time-complexity $t(k)$, makes one query to its lr-encryption oracle, and at most $q(k)$ queries to its decryption oracle. ▌

This immediately implies the following:

Corollary 3.5 [Polynomial-security of the encryption based ID scheme in the CR1 setting] Let $\mathcal{AE}$ be an asymmetric encryption scheme, let $vcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 5. If $\mathcal{AE}$ is polynomially-secure against chosen-ciphertext attack then $\mathcal{ID}$ is polynomially-secure in the CR1 setting. ▌

3.3 An identification based protocol

IDENTIFICATION BASED PROTOCOL. As discussed in the introduction, proof of knowledge based

identification protocols of the Fiat-Shamir type cannot be secure against reset attacks. In this section, however, we present a general transformation of such identification schemes into secure ones in the CR1 setting. We start with identification schemes that consists of three moves, an initial commitment COM of the prover, a random value CH_V, the challenge, of the verifier and a conclusive response RESP from the prover. We call a protocol obeying this structure a canonical identification scheme.

Loosely speaking, we will assume that the underlying canonical identical scheme $\mathcal{CID}$ is secure against non-resetting attacks in the CR1 model, i.e., against attacks where the adversary merely runs concurrent sessions with the prover without resets before engaging in a verification. In addition to the Fiat-Shamir system [20], most of the well-known practical identification schemes also achieve this security level, for example Ong-Schnorr [29, 32] for some system parameters, Okamoto-Guillou-Quisquater [25, 28] and Okamoto-Schnorr [31, 28]. Nonetheless, there are also protocols which are only known to be secure against sequential attacks (e.g. [33]).

To avoid confusion with the derived scheme $\mathcal{ID}$, instead of writing $\mathsf{Send}(\mathsf{prvmsg},\ldots)$ and $\mathsf{Send}(\mathsf{vfmsg},\ldots)$, we denote the algorithms generating the commitment, challenge and response message for the CID-protocol $\mathcal{CID}$ by $\mathcal{CID}(\mathsf{cmt},\ldots)$, $\mathcal{CID}(\mathsf{chall},\ldots)$, and $\mathcal{CID}(\mathsf{resp},\ldots)$, respectively, and the verification step by $\mathcal{CID}(\mathsf{vf},\ldots)$. We also write $\mathbf{Adv}^{\text{id-nr-cr1}}_{\mathcal{CID},I_{\text{CID}}}(k)$ for the probability that an impersonator I_{CID} succeeds in an attack on scheme $\mathcal{CID}$ in the non-resetting CR1 setting.

PROTOCOL AND SECURITY. Our solution originates from the work of [11] about resettable zero-knowledge. In order to ensure that the adversary does not gain any advantage from resetting the prover, we insert a new first round into the CID-identification protocol in which the verifier non-interactively commits to his challenge CH_V. The parameters for this commitment scheme become part of the public key. This keeps the adversary from resetting the prover to the challenge-message and completing the protocol with different challenges.

In addition, we let the prover determine the random values in his identification by applying a pseudorandom function to the verifier's initial commitment. Now, if the adversary resets the prover (with the same random tape) to the outset of the protocol and commits to a different challenge then the prover uses virtually independent randomness for this execution, although having the same random tape. On the other hand, using pseudorandom values instead of truly random coins does not weaken the original identification protocol noticeably. Essentially, this prunes the CR1 adversary into a non-resetting one concerning executions with the prover.

In order to handle the intrusion try we use use a special, so-called trapdoor commitment scheme $\mathcal{TDC}$ for the verifier's initial commitment. This means that there is a secret information such that knowledge of this secret allows to generate a dummy commitment and to find a valid opening to any value later on. Furthermore, the dummy commitment and the fake decommitment are identically distributed to an honestly given commitment and opening to the same value. Without knowing the secret a commitment is still solidly binding. Trapdoor commitment schemes exist under standard assumptions like the intractability of the discrete-log or the RSA or factoring assumption [10] and thus under the same assumptions that the aforementioned CID-identification protocols rely on.

Basically, a trapdoor commitment enables us to reduce an intrusion try of an impersonator I in the derived scheme $\mathcal{ID}$ to one for the CID-protocol. If I initiates a session with the verifier in $\mathcal{ID}$ then we can first commit to a dummy value $0^{vcl(k)}$ without having to communicate with the verifier in $\mathcal{CID}$. When I then takes the next step by sending COM, we forward this commitment to our verifier in $\mathcal{CID}$ and learn the verifier's challenge. Knowing the secret key sk_{TDC} for the trapdoor scheme we can then find a valid opening for our dummy commitment with respect to the challenge. Finally, we forward I's response in our attack.

The scheme is displayed in Figure 6. See Appendix B.3 and B.4 for definitions and notions. The discussion above indicates that any adversary I for $\mathcal{ID}$ does not have much more power than a non-resetting impersonator attacking $\mathcal{CID}$ and security of $\mathcal{ID}$ follows from the security of $\mathcal{CID}$.

Theorem 3.6 [Concrete security of the identification based scheme in the CR1 setting] Let $\mathcal{CID}$ be an CID-identification protocol and let $vcl(\cdot)$ be a polynomially-bounded function. Also, let $\mathcal{PRF}$ be a pseudorandom function family and denote by $\mathcal{TDC}$ a trapdoor commitment scheme. Let $\mathcal{ID}$ be the associated identification scheme as per Figure 6. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR1 setting then there exists an adversary I_{CID} attacking $\mathcal{CID}$ in a non-resetting CR1 attack such that

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) \leq q(k) \cdot \mathbf{Adv}^{\mathcal{PRF}}_{(t,q)}(k) + \mathbf{Adv}^{\mathcal{TDC}}_{t}(k) + \mathbf{Adv}^{\text{id-nr-cr1}}_{\mathcal{CID},I_{\mathrm{CID}}}(k)\ . \tag{3}$$

Furthermore I_{CID} has time-complexity $t(k)$ and runs at most $q(k)$ sessions with the prover before trying to intrude.

As usual we have:

Corollary 3.7 [Polynomial-security of the identification based scheme in the CR1 setting] Let $\mathcal{PRF}$ be a polynomially-secure pseudorandom function family and let $\mathcal{TDC}$ be a polynomially-secure trapdoor commitment scheme, set $vcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 6. If $\mathcal{CID}$ is a polynomially-secure CID-identification protocol in the non-resetting CR1 model then $\mathcal{ID}$ is polynomially-secure in the CR1 setting. ▌

Note that the public key in our CR1-secure identification scheme consists of two independent parts, pk_{CID} and pk_{TDC}. For concrete schemes the key generation may be combined and simplified. For instance, for Okamoto-Schnorr the public key of the identification protocol describes a group of prime order q, two generators g_1, g_2 of that group and the public key $X = g_1^{x_1} g_2^{x_2}$ for secret $x_1, x_2 \in \mathsf{Z}_q$. The prover sends $\textsc{com} = g_1^{r_1} g_2^{r_2}$ and replies to the challenge $\textsc{ch}_V$ by transmitting $y_i = r_i + \textsc{ch}_V x_i \bmod q$ for $i = 1, 2$. In this case, the public key for the trapdoor commitment scheme could be given by $g_1, g_3 = g_1^z$ for random trapdoor $z \in \mathsf{Z}_q$, and the commitment function maps a value c and randomness R_c to $g_1^c g_3^{R_c}$.

3.4 A zero-knowledge based protocol

As we discussed in the Introduction the idea of [18] of proving identity by employing a zero knowledge proof of knowledge has been the accepted paradigm for identification protocols in the smartcard setting. Unfortunately, as we indicated, in the resettable setting this paradigm cannot work.

Resettable Zero Knowledge Based Identity. We thus instead propose the following paradigm. Let L be a hard NP language for which there is no known efficient procedures for membership testing but for which there exists a randomized generating algorithm G which outputs pairs (x, w), where $x \in L$ and w is an NP-witness that $x \in L$. (The distribution according to which (x, w) is generated should be one for which it is hard to tell whether $x \in L$ or not). Each user Alice will run G to get a pair (x, w) and will then publish x as its public key. To prove her identity Alice will run a resettable zero-knowledge proof that $x \in L$.

Protocol. To implement the above idea we need resettable zero-knowledge proofs for L. For this we turn to the work of [11]. In [11] two resettable zero-knowledge proofs for any NP language are proposed: one which takes a non-constant number of rounds and works against a computationally unbounded prover, and one which only takes a constant number of rounds and works against computationally bounded provers (i.e argument) and requires the verifiers to have published public-keys which the prover can access. We propose to utilize the latter, for efficiency sake. Thus, to implement the paradigm, we require both prover and verifier to have public-keys accessible by each other. Whereas the prover's public key is x whose membership in L it will prove to the verifier, the verifier's public key in [11] is used for specifying a perfectly private computationally binding commitment scheme which

the prover must use during the protocol. (Such commitment schemes exist based for example on the strong hardness of Discrete Log Assumption.)

SECURITY. We briefly outline how to prove that the resulting ID protocol is secure in the CR1 setting. Suppose not, and that after launching a CR1 attack, an imposter can now falsely identify himself with a non-negligible probability. Then, we will construct a polynomial time algorithm A to decide membership in L. On input x, A first launches the off-line resetting attack using x as the public key and the simulator – which exists by the zero-knowledge property – to obtain views of the protocol execution. (This requires that the simulator be black-box, but this is true in the known protocols.) If $x \in L$, this view should be identical to the view obtained during the real execution, in which case a successful attack will result, which is essentially a way for A to find a language membership proof. If x not in L, then by the soundness property of a zero-knowledge proof, no matter what the simulator outputs, it will not be possible to prove membership in L.

Acknowledgments

The second author thanks Ran Canetti for discussions about resettable security.

References

[1] M. BELLARE, A. BOLDYREVA AND S. MICALI, "Public-key encryption in a multi-mser Setting: Security proofs and improvements," *Advances in Cryptology – EUROCRYPT '00*, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed., Springer-Verlag, 2000.

[2] M. BELLARE, R. CANETTI, AND H. KRAWCZYK, "Pseudorandom functions revisited: The cascade construction and its concrete security," *Proceedings of the 37th Symposium on Foundations of Computer Science*, IEEE, 1996.

[3] M. BELLARE, R. CANETTI, AND H. KRAWCZYK, "A modular approach to the design and analysis of authentication and key exchange protocols," *Proceedings of the 30th Annual Symposium on the Theory of Computing*, ACM, 1998.

[4] M. BELLARE, A. DESAI, D. POINTCHEVAL AND P. ROGAWAY, "Relations among notions of security for public-key encryption schemes," *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[5] M. BELLARE, M. FISCHLIN, S. GOLDWASSER AND S. MICALI, "Identification protocols secure against reset attacks," Preliminary version of this paper, *Advances in Cryptology – EUROCRYPT '01*, Lecture Notes in Computer Science Vol. 2045, B. Pfitzmann ed., Springer-Verlag, 2001.

[6] M. BELLARE AND O. GOLDREICH, "On defining proofs of knowledge," *Advances in Cryptology – CRYPTO '92*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[7] M. BELLARE AND S. MICALI, "How to sign given any trapdoor permutation," *JACM*, Vol. 39, No. 1, January 1992, pp. 214–233.

[8] M. BELLARE, D. POINTCHEVAL AND P. ROGAWAY, "Authenticated key exchange secure against dictionary attack," *Advances in Cryptology – EUROCRYPT '00*, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed., Springer-Verlag, 2000.

[9] M. BELLARE AND P. ROGAWAY, "Entity authentication and key distribution", *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[10] G. BRASSARD, D. CHAUM AND C. CRÉPEAU, "Minimum Disclosure Proofs of Knowledge," *Journal of Computer and Systems Science*, Vol. 37, No. 2, 1988, pp. 156–189.

[11] R. CANETTI, S. GOLDWASSER, O. GOLDREICH AND S. MICALI, "Resettable zero-knowledge," *Proceedings of the 32nd Annual Symposium on the Theory of Computing*, ACM, 2000.

[12] R. Cramer and I. Damgård, "New generation of secure and practical RSA-based signatures," *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[13] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen-ciphertext attack," *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[14] R. Cramer and V. Shoup, "Signature schemes based on the strong RSA assumption," *Proceedings of the 6th Annual Conference on Computer and Communications Security*, ACM, 1999.

[15] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography", *SIAM J. on Computing*, Vol. 30, No. 2, 2000, pp. 391–437. Preliminary version in STOC 91.

[16] C. Dwork and M. Naor, "An efficient existentially unforgeable signature scheme and its applications," *J. of Cryptology*, Vol. 11, No. 3, 1998, pp. 187–208.

[17] C. Dwork, M. Naor and A. Sahai, "Concurrent zero-knowledge," *Proceedings of the 30th Annual Symposium on the Theory of Computing*, ACM, 1998.

[18] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity," *J. of Cryptology*, Vol. 1, 1988, pp. 77-94.

[19] U. Feige and A. Shamir, "Witness indistinguishable and witness hiding protocols," *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, ACM, 1990.

[20] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Cryptology – CRYPTO '86*, Lecture Notes in Computer Science Vol. 263, A. Odlyzko ed., Springer-Verlag, 1986.

[21] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217.

[22] S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. on Computing*, Vol. 18, No. 1, pp. 186–208, February 1989.

[23] S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal of Computing*, Vol. 17, No. 2, April 1988, pp. 281–308.

[24] R. Gennaro, S. Halevi and T. Rabin, "Secure hash-and-sign signatures without the random oracle," *Advances in Cryptology – EUROCRYPT '99*, Lecture Notes in Computer Science Vol. 1592, J. Stern ed., Springer-Verlag, 1999.

[25] L.C. Guillou and J.-J. Quisquater, "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory," *Advances in Cryptology – EUROCRYPT '88*, Lecture Notes in Computer Science Vol. 330, C. Gunther ed., Springer-Verlag, 1988.

[26] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proceedings of the 21st Annual Symposium on the Theory of Computing*, ACM, 1989.

[27] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, ACM, 1990.

[28] T. Okamoto, "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes," *Advances in Cryptology – CRYPTO '92*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[29] H. Ong and C.P. Schnorr, "Fast Signature Generation with a Fiat-Shamir Identification Scheme" *Advances in Cryptology – EUROCRYPT '90*, Lecture Notes in Computer Science Vol. 473, I. Damgård ed., Springer-Verlag, 1990.

[30] T.P. Pedersen, "Non-Interactive and Information-Theoretical Secure Verifiable Secret Sharing," *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[31] C.P. Schnorr, "Efficient Signature Generation by Smart Cards," *J. of Cryptology*, Vol. 4, 1991, pp. 161–174.

[32] C.P. SCHNORR, "Security of 2^t-Root Identification and Signatures" *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[33] V. SHOUP, "On the Security of a Practical Identification Scheme," *J. of Cryptology*, Vol. 12, 1999, pp. 247–260.

[34] C. RACKOFF AND D. SIMON, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[35] J. ROMPEL, "One-Way Functions are Necessary and Sufficient for Secure Signatures," *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, ACM, 1990.

A Remarks about the notions of security

THE IDENTIFICATION PROBLEM BEING CONSIDERED. We are considering unilateral identification. (One party, the prover, wants to identify itself to another party, the verifier. The other possibility is multilateral identification in which both parties want to identify themselves to each other.) We are in a public-key setting, also called the asymmetric setting. (The prover's public key is known to the verifier. Other possibilities are that the identification is based on shared keys, also called the symmetric setting, or involves a trusted authentication server, the so-called three party setting.) In some contexts —notably that of authenticated session-key exchange in a concurrent setting— the identification problem has been called the entity authentication problem. It is the same problem.

IDENTIFICATION AS A PRELUDE TO SECURE SESSIONS AND THE ROLE OF SESSION KEYS. Identification is hardly an end in itself: an entity goes through an identification process in order to then conduct some transaction that is allowed only to this entity. For example, you first identify yourself to the ATM machine and then withdraw cash. As this example indicates we imagine the transaction as an exchange between prover and verifier taking place after the verifier has accepted in the identification protocol. In the smartcard setting (setting one) this picture is valid because once identification is completed, an adversary cannot step in. (Your card is in the ATM machine and until it is removed the adversary is cut off.) In the Internet setting (setting two) however, identification by itself is largely useless because an adversary can "hijack" the ensuing session, meaning impersonate the prover in the transaction flows that follow the identification, by simply waiting for the verifier to accept and then sending its own messages to the verifier. To have secure transactions, some information from the identification process must be used to authenticate flows in the transaction. This information is usually a session key. Identification without session key exchange is for practical purposes hardly useful in setting two, which is why previous works such as [9, 8] have looked at the problems in combination. In this paper however our focus is the new issues raised by reset attacks and in order to get a better understanding of them in setting two we simplify by decoupling the identification and the session key exchange. Our protocols can be modified to also distribute a session key.

THE NEED FOR MULTIPLE PROVER INSTANCES. Could we simplify the model by providing only a single prover-instance oracle? The answer is no. We can give an example protocol that is secure if the adversary can access only a single prover instance, but is insecure if the adversary can access polynomially-many prover instances.

B Primitives used and their security

Our protocols make use of signature schemes satisfying some special properties, and of standard chosen-ciphertext secure encryption schemes. This section recalls the necessary background.

$(pk, sk) \leftarrow \mathcal{DS}(\mathsf{keygen}, k)$ — Generate public key pk and matching secret key sk
$\text{SIG} \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{MSG})$ — Compute signature of message MSG
$\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{vf}, pk, \text{MSG}, \text{SIG})$ — Verify that SIG is a valid signature of MSG (accept or reject)

Figure 7: The *digital signature scheme description* $\mathcal{DS}$ describes all functionalities associated to the signature scheme.

B.1 Stateless digital signature schemes

SIGNATURE SCHEMES. A digital signature scheme is specified by a description function $\mathcal{DS}$, which, as indicated in Figure 7, specifies how keys are generated, how messages are signed, and how candidate signatures are verified. (As usual it is required that true signatures —meaning those generated by $\mathcal{DS}(\mathsf{sign}, sk, \cdot)$— always successfully pass the verification test.) The key generation algorithm is probabilistic and the verification algorithm is deterministic. The signature algorithm merits a separate discussion which will come later.

SECURITY OF A SIGNATURE SCHEME. The usual definition of security against chosen-message attack is adopted [23].

Definition B.1 [Security of a digital signature scheme] Let $\mathcal{DS}$ be a digital signature scheme description, F an adversary (called a *forger* in this context) having access to an oracle, and k the security parameter. Define

$\mathbf{Experiment}^{\text{ds}}_{\mathcal{DS},F}(k)$
$(pk, sk) \leftarrow \mathcal{DS}(\mathsf{keygen}, k)$; $\text{WIN}_F \leftarrow \mathsf{false}$
$(\text{MSG}, \text{SIG}) \leftarrow F^{\mathcal{DS}(\mathsf{sign}, sk, \cdot)}(pk)$
`If` $\mathcal{DS}(\mathsf{vf}, pk, \text{MSG}, \text{SIG}) = \mathsf{accept}$ and F never made oracle query MSG
  `then` $\text{WIN}_F \leftarrow \mathsf{true}$

The *advantage* of forger F is

$$\mathbf{Adv}^{\text{ds}}_{\mathcal{DS},F}(k) = \Pr[\,\text{WIN}_F = \mathsf{true}\,]$$

where the probability is with respect to $\mathbf{Experiment}^{\text{ds}}_{\mathcal{DS},F}(k)$. Digital signature scheme $\mathcal{DS}$ is said to be *polynomially-secure* if $\mathbf{Adv}^{\text{ds}}_{\mathcal{DS}}(\cdot)$ is negligible for any forger F of time-complexity polynomial in k. ∎

The time-complexity $t(k)$ of adversary F is defined as the execution time of $\mathbf{Experiment}^{\text{ds}}_{\mathcal{DS},F}(k)$, as with previous definitions.

STATE AND RANDOMIZATION IN SIGNING. The signing algorithm $\mathcal{DS}(\mathsf{sign}, sk, \cdot)$ might be stateful (and possibly randomized); randomized but not stateful; or deterministic and stateless. We label a scheme in this regard according to the attribute of its signing algorithm, meaning the scheme is referred to as stateful (resp. stateless, randomized, deterministic) if the signing algorithm is stateful (resp. stateless, randomized, deterministic). The difference is important to the application to identification so we detail it. In a stateful scheme —this is called "history dependent" in some works [23]— the signer maintains some state information state across invocations of the signing procedure. When a message is received, the signer flips some coins; then produces a signature as a function of state, the coins flipped, the message and the keys; then updates state as a function of the coins and message; finally stores state so that it is available at the next invocation of the signing procedure. In a randomized but stateless scheme, the signing algorithm flips coins upon each invocation, but no global state is maintained across invocations. In the simplest case the signing algorithm is not randomized (ie. deterministic) and not stateful (ie. stateless). It associates to any message a unique signature.

Definition B.1 applies regardless of whether the signing procedure is stateful or stateless, randomized or deterministic. But we stress that the oracle $\mathcal{DS}(\mathsf{sign}, sk, \cdot)$ provided to the forger F in Definition B.1 is responsible for implementing any statefulness or randomization in the signing process and does so as described above. In particular, if the scheme is randomized, *fresh* coins are picked and used upon each invocation of the oracle; if the scheme is stateful, the oracle maintains and updates the state. (In particular the adversary has no way to force the oracle to reuse a particular set of coins for two signatures. This will be important later.)

The basic versions of the schemes of [23, 7, 26, 35] are (randomized and) stateful. The more efficient schemes of [16, 12] are also (randomized and) stateful. Examples of (randomized but) stateless schemes are those of [24, 14]. Although there seem to be few schemes that are "naturally" stateless, deterministic and secure, any signature scheme can be made stateless and deterministic while preserving security. A well-known transformation —attributed in [23] to Goldreich and Levin— transforms a stateful scheme into a (randomized but) stateless one by using a binary tree structure. A stateless signing algorithm can be derandomized —while preserving statelessness and security— via the following (folklore) trick: the secret key is expanded to include a key κ specifying an instance $\mathcal{PRF}(\mathsf{eval}, \kappa, \cdot)$ of a family of pseudorandom functions (see [21] or Appendix B.3), and to sign message MSG compute $R_{\mathrm{MSG}} = \mathcal{PRF}(\mathsf{eval}, \kappa, \mathrm{MSG})$ and use R_{MSG} as the coins f or the signing algorithm. Combining this with Rompel's result [35] implies:

Proposition B.2 If there exists a one-way function then there exists a stateless, determinstic polynomially-secure digital signature scheme.

This addresses the "theoretical" question of the existence of stateless, deterministic signature schemes by indicating they exist under the minimal possible complexity assumption. The next question —on the "practical" side— is about the cost of available solutions. The most efficient known signature schemes that are provably-secure under standard —meaning non-random oracle— assumptions are those of [24, 14]. These schemes are randomized but stateless. Derandomization is cheap if properly implemented: Instantiate the pseudorandom function used in the derandomization process discussed above with a block cipher, and the impact on the cost of signing —already involving public key operations— is negligible. In this way we get efficient, stateless, deterministic signature schemes that are provably polynomially-secure under standard assumptions. (One can also consider the earlier schemes of [16, 12] but they are less efficient than those of [24, 14] and also are stateful. Making a stateful scheme stateless seems to be more costly than derandomizing an already stateless scheme.)

B.2 CCA2-secure Encryption schemes

ENCRYPTION SCHEMES. An asymmetric encryption scheme is specified by a description function $\mathcal{AE}$, which as indicated, in Figure 8, specifies how keys are generated, how messages are encrypted, and how ciphertexts are decrypted. (As usual it is required that if ciphertext CTXT is generated via $\mathcal{AE}(\mathsf{enc}, pk, \mathrm{MSG})$ then $\mathcal{AE}(\mathsf{dec}, pk, sk, \mathrm{CTXT})$ returns MSG.) The key generation and encryption algorithms are probabilistic while the decryption algorithm is deterministic.

SECURITY OF AN ENCRYPTION SCHEME. We require indistinguishability against chosen-ciphertext attack. The version of the definition we adopt, from [1], allows the adversary multiple "test" message pairs rather than a single one, and was shown by them to be polynomially equivalent to the more standard formuation of [34]. Define $\mathsf{LR}(\mathrm{MSG}_0, \mathrm{MSG}_1, b) = \mathrm{MSG}_b$ for any equal-length strings $\mathrm{MSG}_0, \mathrm{MSG}_1$ and bit b.

Definition B.3 [Security of an encryption scheme under chosen-ciphertext attack] Let $\mathcal{AE}$ be an asymmetric encryption scheme description. Let E be an adversary (called an *eavesdropper* in

$(pk, sk) \leftarrow \mathcal{AE}(\mathsf{keygen}, k)$ — Generate public key pk and matching secret key sk
$\text{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \text{MSG})$ — Compute encryption of message MSG
$\text{OUT} \leftarrow \mathcal{AE}(\mathsf{dec}, pk, sk, \text{CTXT})$ — The decryption procedure takes the public key, secret key and a ciphertext CTXT and returns OUT which is either a message MSG or the special symbol $\perp$ to indicate it considered the ciphertext invalid.

Figure 8: The *asymmetric encryption scheme description* $\mathcal{AE}$ describes all functionalities associated to the encryption scheme.

this context) having access to two oracles, the first taking as input any two strings of equal length and the second any string. Let k be the security parameter. Define

$\mathbf{Experiment}^{\text{lr-cca}}_{\mathcal{AE},E}(k)$
$(pk, sk) \leftarrow \mathcal{AE}(\mathsf{keygen}, k)$; $\text{WIN}_E \leftarrow \mathsf{false}$
$\text{CB} \xleftarrow{R} \{0,1\}$ // Random challenge bit //
$\text{GB} \leftarrow E^{\mathcal{AE}(\mathsf{enc},pk,\mathsf{LR}(\cdot,\cdot,\text{CB})),\,\mathcal{AE}(\mathsf{dec},pk,sk,\cdot)}(pk)$ // Eavesdropper's guess bit //
`If` $\text{GB} = \text{CB}$ and $\mathcal{AE}(\mathsf{dec}, pk, sk, \cdot)$ was never called on a ciphertext
returned by $\mathcal{AE}(\mathsf{enc}, pk, \mathsf{LR}(\cdot, \cdot, \text{CB}))$
`then` $\text{WIN}_E \leftarrow \mathsf{true}$

The *advantage* of eavesdropper E is

$$\mathbf{Adv}^{\text{ds}}_{\mathcal{AE},E}(k) = 2 \cdot \Pr[\,\text{WIN}_E = \mathsf{true}\,] - 1$$

where the probability is with respect to $\mathbf{Experiment}^{\text{lr-cca}}_{\mathcal{AE},E}(k)$. Asymmetric encryption scheme $\mathcal{AE}$ is said to be *polynomially-secure* if $\mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE}}(\cdot)$ is negligible for any eavesdropper E of time-complexity polynomial in k. ▌

We call $\mathcal{AE}(\mathsf{enc}, pk, \mathsf{LR}(\cdot, \cdot, \text{CB}))$ the "lr-encryption oracle" where "lr" stands for "left or right."

B.3 Pseudorandom functions

PSEUDORANDOM FUNCTIONS. We keep the definition as simple as possible for our purpose. A pseudorandom function family is a function $\mathcal{PRF}(\mathsf{eval}, \cdot, \cdot)$ in two arguments. The first argument, called the key, has k bits and defines in a straightforward way a function $\mathcal{PRF}(\mathsf{eval}, \kappa, \cdot)$ for any $\kappa \in \{0,1\}^k$. For every $\kappa \in \{0,1\}^k$ the function $\mathcal{PRF}(\mathsf{eval}, \kappa, \cdot)$ has input and output length $\text{inl}(k)$ and $\text{outl}(k)$; the actual choice of $\text{inl}(\cdot)$ and $\text{outl}(\cdot)$ depends on the application.

SECURITY OF A PSEUDORANDOM FUNCTIONS. We adopt the definition of pseudorandom functions being indistinguishable from random functions [21]:

Definition B.4 [Security of a pseudorandom function family] Let $\mathcal{PRF}$ be a pseudorandom function family, D an adversary (called a *distinguisher* in this context) having access to an oracle, and k the security parameter. Define

$\mathbf{Experiment}^{\text{prf-dist}}_{\mathcal{PRF},D}(k, b)$
`If` $b = 0$ `then` $\kappa \xleftarrow{R} \{0,1\}^k$ and let $\mathcal{O}(\cdot) = \mathcal{PRF}(\mathsf{eval}, \kappa, \cdot)$
`If` $b = 1$ `then` let $\mathcal{O}(\cdot)$ be a random function with input/output length $\text{inl}(k)$ and $\text{outl}(k)$
$\text{GB}_b \leftarrow D^{\mathcal{O}(\cdot)}(k)$

The *advantage* of distinguisher D is

$$\mathbf{Adv}^{\text{prf-ind}}_{\mathcal{PRF},D}(k) = |\Pr[\,\text{GB}_1 = 1\,] - \Pr[\,\text{GB}_0 = 1\,]|$$

$(pk, sk) \leftarrow \mathcal{TDC}(\mathsf{keygen}, k)$ — Generate public key pk and secret key sk
$\text{TDCOM} \leftarrow \mathcal{TDC}(\mathsf{cmt}, pk, c; R_c)$ — Compute commitment of value c with randomness R_c
$\mathsf{decision} \leftarrow \mathcal{TDC}(\mathsf{vf}, pk, \text{TDCOM}, c\|R_c)$ — Verify that TDCOM is commitment of c and randomness R_c
$(c', R'_c), \leftarrow \mathcal{TDC}(\mathsf{fake}, sk, c\|R_c, c')$ — Given a value c and randomness R_c and another value c' use the secert key sk to find R'_c such that c', R'_c and c, R_c yield the same commitment

Figure 9: The *trapdoor commitment scheme description* $\mathcal{TDC}$ describes all functionalities associated to the trapdoor commitment scheme.

where the probabilities are with respect to $\mathbf{Experiment}^{\text{prf-dist}}_{\mathcal{PRF},D}(k, b)$. The time-complexity $t(k)$ of D is defined as the maximum execution time $\mathbf{Experiment}^{\text{prf-dist}}_{\mathcal{PRF},D}(k, b)$ for $b = 0, 1$, and the query-complexity is the maximum number of queries D makes to the oracle in either experiment. Set $\mathbf{Adv}^{\mathcal{PRF}}_{(t,q)}(k)$ to be the maximum $\mathbf{Adv}^{\text{prf-ind}}_{\mathcal{PRF},D}(k)$ over all distinguishers D with time-complexity $t(k)$ and query-complexity $q(k)$. The pseudorandom function family $\mathcal{PRF}$ is called *polynomially-secure* if $\mathbf{Adv}^{\text{prf-ind}}_{\mathcal{PRF},D}(\cdot)$ is negligible for any distinguisher D of time-complexity polynomial in k. ▌

B.4 Trapdoor commitments

TRAPDOOR COMMITMENT SCHEMES. A (non-interactive) trapdoor commitment scheme is defined by a function $\mathcal{TDC}$ as displayed in Figure 9. The function $\mathcal{TDC}$ specifies a key generation algorithm, a commitment algorithm, a verification function deciding the correctness of a given commitment, and a faking algorithm that allows to open a commitment with any value c' given the secret key. We demand that a commitment and such a faked opening is identically distributed to a commitment with the correct opening for the same value c'. In particular, this implies that the commitment scheme provides perfect secrecy, i.e., a commitment is distributed independently of the actual value.

SECURITY OF A TRAPDOOR COMMITMENT SCHEME. We require that it is infeasible to find a commitment and ambiguous decommitments.

Definition B.5 [Security of a trapdoor commitment scheme] Let $\mathcal{TDC}$ be a trapdoor commitment scheme description. Let C be an adversary (called a *collision-finder* in this context) and let k be the security parameter. Set

$\mathbf{Experiment}^{\text{tdc-coll}}_{\mathcal{TDC},C}(k)$
$(pk, sk) \leftarrow \mathcal{TDC}(\mathsf{keygen}, k)$; $\text{WIN}_C \leftarrow \mathsf{false}$
$(\text{TDCOM}, c\|R_c, c'\|R'_c) \leftarrow C(k, pk)$
`If` $\mathcal{TDC}(\mathsf{vf}, pk, \text{TDCOM}, c\|R_c) = \mathcal{TDC}(\mathsf{vf}, pk, \text{TDCOM}, c'\|R'_c) = \mathsf{accept}$ and $c \neq c'$
  `then` $\text{WIN}_C \leftarrow \mathsf{true}$

The *advantage* of the collision-finder C is

$$\mathbf{Adv}^{\text{tdc-coll}}_{\mathcal{TDC},C}(k) = \Pr[\,\text{WIN}_C = \mathsf{true}\,]$$

where the probability is with respect to $\mathbf{Experiment}^{\text{tdc-coll}}_{\mathcal{TDC},C}(k)$. The trapdoor commitment scheme is said to be *polynomially-secure* if $\mathbf{Adv}^{\text{tdc-coll}}_{\mathcal{TDC},C}(\cdot)$ is negligible for any eavesdropper C of time-complexity polynomial in k. Set $\mathbf{Adv}^{\mathcal{TDC}}_{t}(k)$ to be the maximum $\mathbf{Adv}^{\text{tdc-coll}}_{\mathcal{TDC},C}(k)$ over all collision-finder C with running time $t(k)$. ▌

ID-BASED TRAPDOOR COMMITMENT SCHEME. For an ID-based trapdoor commitment scheme the key generation algorithm returns a uniformly distributed string $\mathsf{sid}_{\text{TDC}} \in \{0,1\}^{vcl(k)}$ as part of the

secret key. Yet, the public key is distributed independently of this string $\mathsf{sid}_{\mathrm{TDC}}$. The commitment algorithm $\mathcal{TDC}(\mathsf{cmt}, pk, \cdot)$ now takes as input a string $\mathsf{sid} \in \{0,1\}^{vcl(k)}$, a value c and randomness R_c and returns a commitment.

Security for ID-based trapdoor commitment schemes is defined with respect to a collision-finder that gets k, pk and sk (including $\mathsf{sid}_{\mathrm{TDC}}$) as input and is considered to win if it outputs a commitment with valid openings for two different values c, c' and the same sid, where sid is different from $\mathsf{sid}_{\mathrm{TDC}}$. In other words, the trapdoor property is tied to $\mathsf{sid}_{\mathrm{TDC}}$ and does not help to overcome the binding property for other IDs.

As an example of an ID-based trapdoor commitment scheme we sketch a solution based on Pedersen's discrete-log commitment scheme [30]; similar solutions can be erected for RSA and factoring. The public key consists of a group of prime order q and two random generators g_1, g_2 of the group, as well as another generator g_3. The latter generator is defined by $g_3 = g_1^{-\mathsf{sid}_{\mathrm{TDC}}} g_2^z$ for random $\mathsf{sid}_{\mathrm{TDC}} \in \{0,1\}^{vcl(k)}$ and random $z \in \mathsf{Z}_q$. Clearly, g_3 hides $\mathsf{sid}_{\mathrm{TDC}}$ information-thereotically.

A commitment to (sid, c, R_c) is defined by $(g_1^{\mathsf{sid}} g_3)^c g_2^{R_c}$. The trapdoor sk_{TDC} equals $\mathsf{sid}_{\mathrm{TDC}}$ and z. Because $g_1^{\mathsf{sid}_{\mathrm{TDC}}} g_3 = g_2^z$ is is easy to adapt a decommitment for $\mathsf{sid}_{\mathrm{TDC}}$ by the discrete-log trapdoor property [10]. Namely, given $c, R_c, \mathsf{sid}_{\mathrm{TDC}}, z$ and c' let $R'_c = zc + R_c - zc' \bmod q$ such that

$$(g_1^{\mathsf{sid}_{\mathrm{TDC}}} g_3)^{c'} g_2^{R'_c} = g_2^{zc'+R'_c} = g_2^{zc+R_c} = (g_1^{\mathsf{sid}_{\mathrm{TDC}}} g_3)^c g_2^{R_c}$$

On the other side, for distinct $c \neq c'$ an ambiguous decommitment (c, R_c), (c', R'_c) for the same $\mathsf{sid} \neq \mathsf{sid}_{\mathrm{TDC}}$ implies

$$(g_1^{\mathsf{sid}} g_3)^c g_2^{R_c} = (g_1^{\mathsf{sid}} g_3)^{c'} g_2^{R'_c}$$

or equivalently,

$$g_1^{(\mathsf{sid}-\mathsf{sid}_{\mathrm{TDC}})(c-c')} = g_2^{(R'_c+zc')-(R_c+zc)}.$$

Since $\mathsf{sid} - \mathsf{sid}_{\mathrm{TDC}}, c - c' \neq 0$ one can easily compute $\log_{g_1} g_2$, contradicting the discrete-log assumption.

C CR2-secure Identification protocols

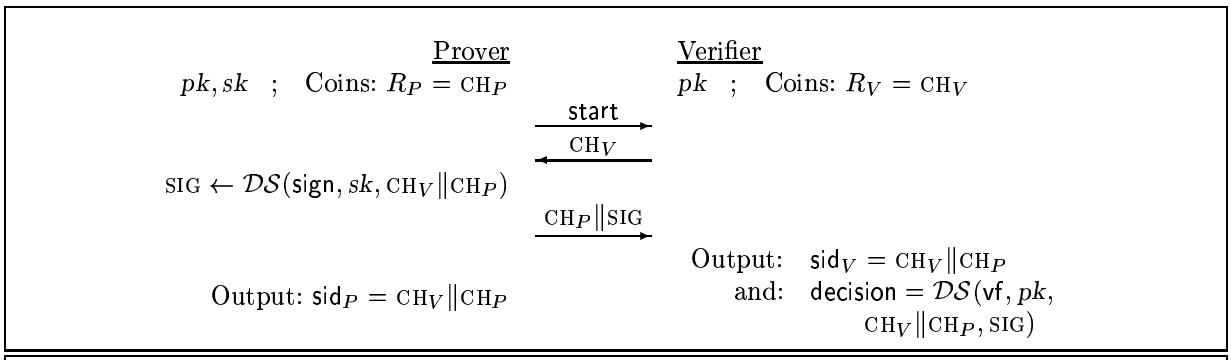
The protocols of Section 3 are not secure in the CR2 setting. We show how the same paradigms can be applied to yield modified protocols that are secure in the CR2 setting.

C.1 A signature based protocol

The signature based protocol of Figure 4 which we proved secure in the CR1 setting is not secure in the CR2 setting, even in the absence of reset attacks, since there are no session ids. Indeed, if an adversary activates two prover instances and plays the role of the verifier with each, then both accept with the same session id, so the adversary wins as per our definition. In fact any identification protocol in which the session ids have length $O(\log k)$ is not polynomially-secure in the CR2 setting.

We modify the protocol of Figure 4 by having the prover select a random "challenge" and sign the concatenation of this with the verifier's challenge. The session id (for both the prover and the verifier) is the concatenation of the two challenges. We will prove that this protocol is secure in the CR2 setting.

PROTOCOL AND SECURITY. Let $\mathcal{DS}$ be a deterministic, stateless signature scheme. Figure 10 illustrates the flows of the associated identification protocol $\mathcal{ID}$ and then provides the protocol description. (The latter includes several checks omitted in the picture but important for security against resets.) Parameters of the protocol are the length $vcl(k)$ of the verifier's random challenge and the length $pcl(k)$ of the prover's random challenge. The random tape, for each party, is its challenge. Refer to



$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{DS}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ has same key generation process as $\mathcal{DS}$

$\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_P)$ where $|R_P| = pcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- `If` $l \notin \{0, 2\}$ `then Return` $\perp$
- `If` $l = 0$ `then Return start`
- `If` $|\text{MSG}_2| \neq vcl(k)$ then `Return` $\perp$
- $\text{CH}_V \leftarrow \text{MSG}_2$; $\text{CH}_P \leftarrow R_P$
- $\text{SIG} \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{CH}_V \| \text{CH}_P)$
- `Return` $\text{CH}_P \| \text{SIG}$

$\mathcal{ID}(\mathsf{prvsid}, sk, x; R_P)$ where $|R_P| = pcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- `If` $l \neq 3$ or $|\text{MSG}_2| \neq vcl(k)$ `then Return` $\perp$
- $\text{CH}_V \leftarrow \text{MSG}_2$; $\mathsf{sid}_P \leftarrow \text{CH}_V \| R_P$
- `Return` sid_P

$\mathcal{ID}(\mathsf{vfmsg}, pk, x; R_V)$ where $|R_V| = vcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- `If` $l \neq 1$ `then Return` $\perp$
- $\text{CH}_V \leftarrow R_V$
- `Return` CH_V

$\mathcal{ID}(\mathsf{vfend}, pk, x; R_V)$ where $|R_V| = vcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- `If` $l \neq 3$ or $\text{MSG}_2 \neq R_V$ `then Return` $\perp$
- Parse MSG_3 as $\text{CH}_P \| \text{SIG}$ with $|\text{CH}_P| = pcl(k)$
- $\text{CH}_V \leftarrow \text{MSG}_2$; $\mathsf{sid}_V \leftarrow \text{CH}_V \| \text{CH}_P$
- $\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{vf}, pk, \text{CH}_V \| \text{CH}_P, \text{SIG})$
- `Return` $(\mathsf{sid}, \mathsf{decision})$

Figure 10: Reset-secure identification protocol $\mathcal{ID}$ for the CR2 setting based on a deterministic, stateless digital signature scheme $\mathcal{DS}$: Schematic followed by full protocol description.

Definition 2.2 and Definition B.1 for the meanings of terms used in the theorem below. The proof is similar to that of Theorem 3.1 and is omitted.

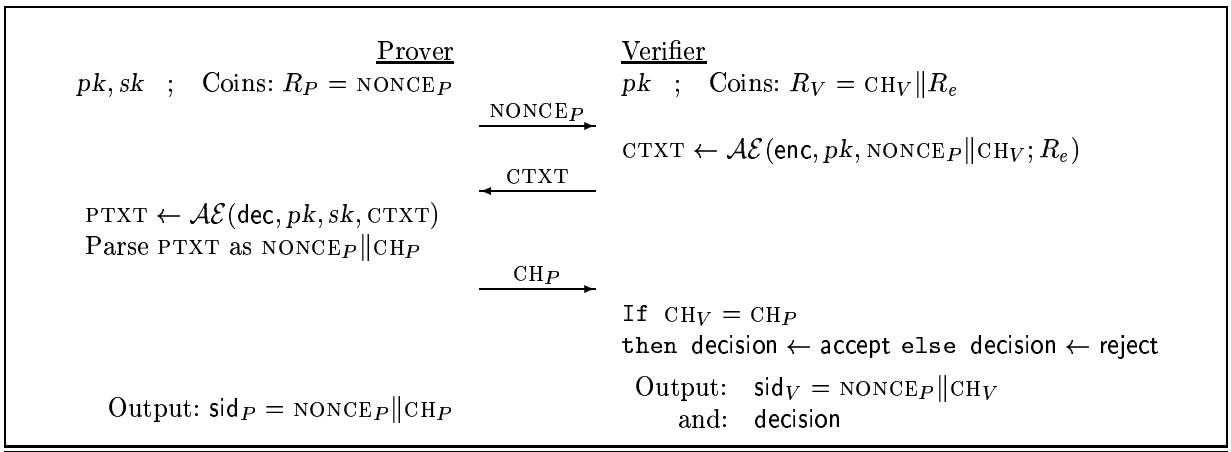
Theorem C.1 [Concrete security of the signature based ID scheme in the CR2 setting] Let $\mathcal{DS}$ be a deterministic, stateless signature scheme, let $vcl(\cdot)$ and $pcl(\cdot)$ be polynomially-bounded functions, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 10. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR2 setting then there exists a forger F attacking $\mathcal{DS}$ such that

$$\mathbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{ds}}_{\mathcal{DS},F}(k) + \frac{q(k)}{2^{vcl(k)}} + \frac{q(k)^2 - q(k)}{2^{pcl(k)+1}} \,. \tag{4}$$

Furthermore F has time-complexity $t(k)$ and makes at most $q(k)$ signing queries in its chosen-message attack on $\mathcal{DS}$. ▌

As before we get two corollaries:

Corollary C.2 [Polynomial-security of the signature based ID scheme in the CR2 setting] Let $\mathcal{DS}$ be a deterministic, stateless signature scheme, let $vcl(k) = pcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 10. If $\mathcal{DS}$ is polynomially-secure then $\mathcal{ID}$ is polynomially-secure in the CR2 setting. ▌



$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{AE}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ has same key generation process as $\mathcal{AE}$

$\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_P)$ where $|R_P| = pcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- If $l \notin \{0, 2\}$ then Return $\perp$
- If $l = 0$ then Return R_P
- $\text{CTXT} \leftarrow \text{MSG}_2$; $\text{PTXT} \leftarrow \mathcal{AE}(\mathsf{dec}, sk, \text{CTXT})$
- If $|\text{PTXT}| \neq pcl(k) + vcl(k)$ then Return $\perp$
- Parse PTXT as $\text{NONCE}_P \| \text{CH}_P$ with $|\text{NONCE}_P| = pcl(k)$ and $|\text{CH}_P| = vcl(k)$
- If $\text{NONCE}_P \neq R_P$ then Return $\perp$
- Return CH_P

$\mathcal{ID}(\mathsf{prvsid}, sk, x; R_P)$ where $|R_P| = pcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- If $l \neq 3$ then Return $\perp$
- $\text{CH}_P \leftarrow \text{MSG}_3$; $\mathsf{sid} \leftarrow R_P \| \text{CH}_P$
- If $|\text{CH}_P| \neq vcl(k)$ then Return $\perp$
- Return sid

$\mathcal{ID}(\mathsf{vfmsg}, pk, x; R_V)$
- Parse R_V as $\text{CH}_V \| R_e$ with $|\text{CH}_V| = vcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- If $l \neq 1$ then Return $\perp$
- If $|\text{MSG}_1| \neq pcl(k)$ then Return $\perp$
- $\text{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \text{MSG}_1 \| \text{CH}_V; R_e)$
- Return CTXT

$\mathcal{ID}(\mathsf{vfend}, pk, x; R_V)$
- Parse R_V as $\text{CH}_V \| R_e$ with $|\text{CH}_V| = vcl(k)$
- Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$
- If $l \neq 3$ then Return $\perp$
- If $|\text{MSG}_1| \neq pcl(k)$ then Return $\perp$
- $\mathsf{sid} \leftarrow \text{MSG}_3 \| \text{CH}_V$
- If $\text{MSG}_3 = \text{CH}_V$ then decision ← accept else decision ← reject
- Return (sid, decision)

Figure 11: Reset-secure identification protocol $\mathcal{ID}$ for the CR2 setting based on a chosen-ciphertext attack secure asymmetric encryption scheme $\mathcal{AE}$: Schematic followed by full protocol description.

Corollary C.3 [Existence of an ID scheme polynomially-secure in the CR2 setting] Assume there exists a one-way function. Then there exists an identification scheme that is polynomially-secure in the CR2 setting.

C.2 An encryption based protocol

The encryption based protocol of Figure 5 (which we proved secure in the CR1 setting) does not have session ids, so the discussion above implies that it is not secure in the CR2 setting. Modifying this protocol to make it secure in the CR2 setting is more tricky than in the case of the signature based protocol. The first thought is to have the prover pick some random challenge CH_P and convey it, in the clear, along with PTXT. Both parties then set their session id to $\text{CH}_P \| \text{PTXT}$. But this protocol is insecure. An adversary can modify CH_P after the prover sends it, and the verifier would still accept, but with a session id not shared by any prover instance, so that the adversary wins. (Modification

of CH_P by the verifier in the protocol of Figure 10 would lead to the verifier rejecting because of the attached signature, but we do not want to use signatures here.) Instead we have the prover send a nonce (random string) in its first move, and have the verifier encrypt the concatenation of the prover and verifier challenges.

PROTOCOL AND SECURITY. Let $\mathcal{AE}$ be an asymmetric encryption scheme polynomially-secure against chosen-ciphertext attack. Figure 11 illustrates the flows of the associated identification protocol $\mathcal{ID}$ and then provides the protocol description. Parameters of the protocol are the length $vcl(k)$ of the verifier's random challenge and the length $pcl(k)$ of the prover's random challenge. The random tape of the prover is its nonce, and that of the verifier is its challenge together with coins R_e sufficient for one invokation of the encryption algorithm. Refer to Definition 2.2 and Definition B.3 for the meanings of terms used in the theorem below. The proof is similar to that of Theorem 3.4 and is omitted.

Theorem C.4 [Concrete security of the encryption based ID scheme in the CR2 setting] Let $\mathcal{AE}$ be an asymmetric encryption scheme, let $vcl(\cdot)$ and $pcl(\cdot)$ be polynomially-bounded functions, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 11. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR2 setting then there exists an eavesdropper E attacking $\mathcal{AE}$ such that

$$\mathbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + \frac{2q(k)+2}{2^{vcl(k)}} + \frac{q(k)^2 - q(k)}{2^{pcl(k)}} \,. \tag{5}$$

Furthermore E has time-complexity $t(k)$, makes one query to its lr-encryption oracle, and at most $q(k)$ queries to its decryption oracle.

As before we get the corollary:

Corollary C.5 [Polynomial-security of the encryption based ID scheme in the CR2 setting] Let $\mathcal{AE}$ be an asymmetric encryption scheme, let $vcl(k) = pcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 11. If $\mathcal{AE}$ is polynomially-secure against chosen-ciphertext attack then $\mathcal{ID}$ is polynomially-secure in the CR2 setting. ▌

C.3 An identification based protocol

We modify the CR1 secure identification scheme in Section 3.3 to achieve CR2 security. For this we define an adversarial success slightly more stingent: the impersonator is not considered to be victorious anymore if it confuses the verifier and generates session ID collisions in the executions with the prover. Rather, the only way for the adversary to win is by passing the verifier's examination for a fresh session ID. We write $\mathbf{Adv}^{\text{weak-id-cr2}}_{\mathcal{ID},I}(k)$ for the success probability of adversary I winning under this slightly weaker security notion in a CR2-attack against $\mathcal{ID}$.

PROTOCOL AND SECURITY. The key to accomplish CR2-security lies in the extension of the trapdoor commitment scheme to an ID-based one: the key generation algorithm outputs $(pk_{\text{TDC}}, sk_{\text{TDC}})$ such that sk_{TDC} includes a uniformly distributed string $\mathsf{sid}_{\text{TDC}}$ of length $vcl(k)$, and such that pk_{TDC} is distributed independently of $\mathsf{sid}_{\text{TDC}}$. The input to the commitment function takes an additional string of length $vcl(k)$. Given a commitment involving $\mathsf{sid}_{\text{TDC}}$ it is easy to open this commitment with any value later. But it is still infeasible to find ambiguous decommitments for a commitment with $\mathsf{sid} \neq \mathsf{sid}_{\text{TDC}}$, even if one knows sk_{TDC}. An example based on the discrete logarithm is given in Section B.4.

Roughly, an ID-based trapdoor commitment schemes links a session ID to the trapdoor property. So if we simulate the adversary I to derive an impersonater for I_{CID}, as done in the CR1 setting, we can later use the previously generated $\mathsf{sid}_{\text{TDC}}$ in the adversary's intrusion attempt. This means that the adversary cannot use this session ID in its executions with the prover (otherwise the adversary

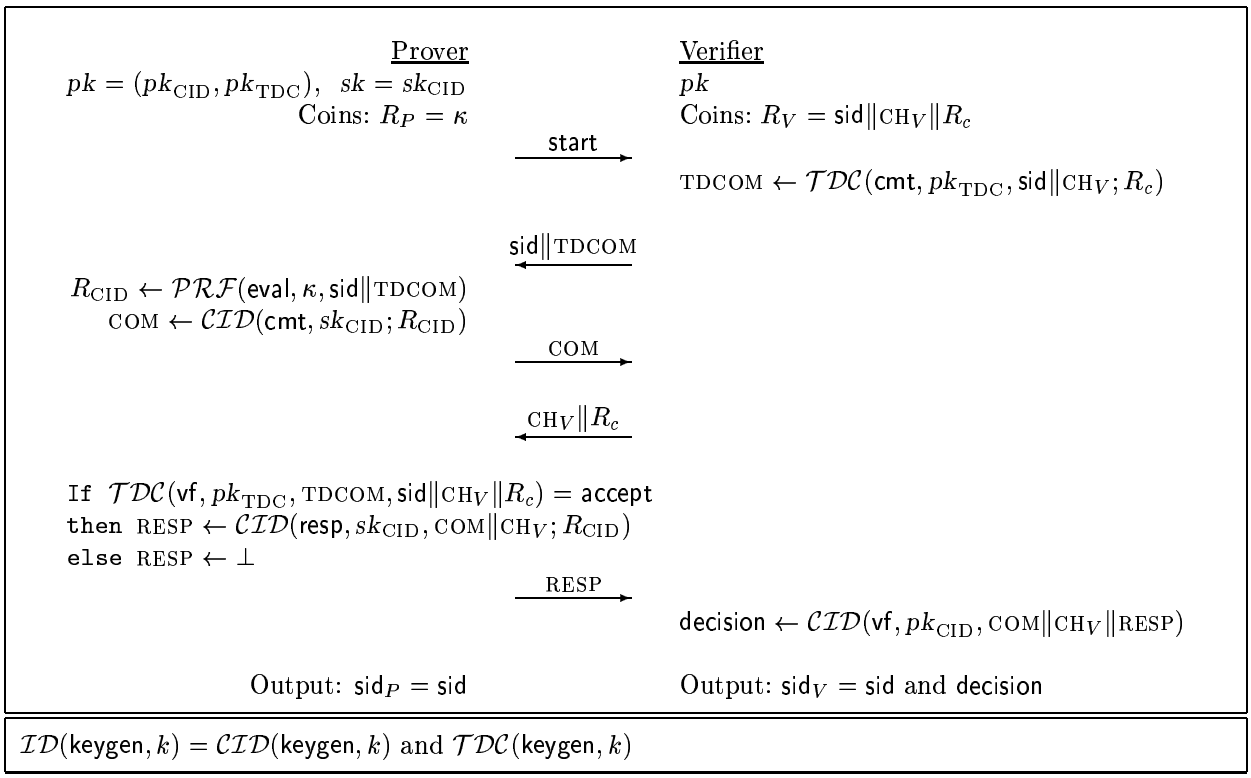


Figure 12: Reset-secure identification protocol $\mathcal{ID}$ for the CR2 setting based on secure CID-identification scheme

is not considered victorious according to the definition). But if the impersonator forgos using $\mathsf{sid}_{\mathrm{TDC}}$ then all its commitments for other session IDs are binding and a similar argument to the one in the CR1 model applies. Since the public key of the trapdoor scheme hides $\mathsf{sid}_{\mathrm{TDC}}$ perfectly, we can later claim that the verifier has chosen $\mathsf{sid}_{\mathrm{TDC}}$ only then.

The difference to the CR1 setting is that the impersonator I may now interleave the execution with the verifier and the ones with prover. Let $\mathbf{Adv}^{\text{id-nr-cr2}}_{\mathcal{CID},I_{\mathrm{CID}}}(k)$ be the success probability of I_{CID} breaking $\mathcal{CID}$ in a non-resetting CR2-attack. Although CID-protocols fail to be secure against such attacks in general, e.g., the woman-in-the-middle adversary breaks such schemes in this setting, luckily they remain secure under a certain condition on the adversary. Therefore, we will still able to start with the previously mentioned known CID-protocols.

To specify the condition under which CID-schemes remain secure, consider an execution of an impersonator I_{CID} attacking $\mathcal{CID}$ in a non-resetting CR2 attack. At some step the verifier sends a random challenge CH_V to I_{CID} and the adversary then finishes the attack, either successfully or not. Define a *challenge reset* to be the following action: reset the state of the prover, the adversary and the verifier to the point before sending CH_V; then transmit another random CH'_V instead and continue the adversary's attack on this new challenge. The reason for considering such challenge-resets is that they are normally used to prove security for CID schemes, refer to [18] for details.

Next we look at what happens on the prover's side in challenge resets. We are especially interested in executions in which the prover has sent a commitment COM before the adversary received CH_V, and in which the impersonator has answered with some challenge CH in that execution with the prover after receiving CH_V. This implies that after a challenge reset the adversary may now decide to send a different challenge CH' instead of CH. We say that the impersonator never finishes an execution with the prover ambiguously if this never happens. For a function $chr(\cdot)$ we say that an CID-identification protocol is *chr-challenge-resettable for* I_{CID} if the impersonator I_{CID} never finishes an execution with

the prover ambiguously, even if $chr(k)$ challenge resets occur. As for the asymptotic behavior, it is understood that a polynomially-secure CID-protocol in the non-resetting CR2 setting refers to security against any polynomially-bounded, non-resetting CR2-adversary $I_{\rm CID}$ for which the protocol is chr-challenge-resettable for any polynomial $chr(\cdot)$.

To clarify the notion we consider two examples. No CID-scheme is even 2-challenge-resettable for the woman-in-the-middle adversary. The reason is such an adversary duplicates all messages of the prover and the verifier and if we execute a challenge reset then the adversary imitates this, too. In contrast, for any non-resetting CR1-adversary any CID-protocol is challenge-resettable because the executions with the prover are already finished when the intrusion try starts.

In comparison to the CR1-secure scheme, here the verifier chooses a random session ID and the ID-based trapdoor scheme is applied to commit to the session ID and the challenge at the beginning of an execution. The session ID is also transmitted in clear together with the commitment. Except for this modified commitment the rest of the protocol remains unchanged. The common session ID is set to the verifier's choice (and thus it is easy for the adversary to make sessions with the prover end up with the same ID).

Theorem C.6 [Concrete security of the identification based scheme in the CR2 setting] Let $\mathcal{CID}$ be an CID-identification protocol and let $chr(\cdot)$, $vcl(\cdot)$ be polynomially-bounded functions. Also, let $\mathcal{PRF}$ be a pseudorandom function family and denote by $\mathcal{TDC}$ an ID-based trapdoor commitment scheme. Let $\mathcal{ID}$ be the associated identification scheme as per Figure 12. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR2 setting then there exists an adversary $I_{\rm CID}$ attacking $\mathcal{CID}$ in a non-resetting CR2 attack such that $\mathcal{CID}$ is $chr(\cdot)$-challenge-resettable for $I_{\rm CID}$ and

$$\mathbf{Adv}^{\text{weak-id-cr2}}_{\mathcal{ID},I}(k) \le q(k) \cdot \mathbf{Adv}^{\mathcal{PRF}}_{(t,q)}(k) + \mathbf{Adv}^{\mathcal{TDC}}_{t\cdot chr}(k) + \mathbf{Adv}^{\text{id-nr-cr2}}_{\mathcal{CID},I_{\rm CID}}(k)\ . \qquad (6)$$

For the asymptotic counterpart we have:

Corollary C.7 [Polynomial-security of the identification based scheme in the CR2 setting] Let $\mathcal{PRF}$ be a polynomially-secure pseudorandom function family and let $\mathcal{TDC}$ be a polynomially-secure ID-based trapdoor commitment scheme, set $vcl(k) = k$, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 12. If $\mathcal{CID}$ is a polynomially-secure CID-identification protocol in the non-resetting CR2 setting then $\mathcal{ID}$ is polynomially-secure in the CR2 setting. ▌

D Proofs

D.1 Proof of Theorem 3.1

Figure 13 describes the forging algorithm F attacking $\mathcal{DS}$. It runs I as a subroutine, itself responding to the latter's oracle queries so as to provide a "simulation" of the environment provided to I in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$, and eventually outputs a forgery. The forger is not in possession of the secret key sk which is used by prover instances but can compensate using its access to the signing oracle. Important to the fact that the time-complexity of F is $t(k)$ —the same as that of I— are our conventions under which the time measured pertains to the entire experiment. (In particular the time used by the signing oracle is not an "extra" for the forger since it corresponds to invocations of the signing algorithm by prover instances in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$.) It remains to verify Equation (1).

We claim that the simulation is "perfect" in the sense that from the point of view of I it is in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$. Barring the use of the signing oracle to compute the signatures, the forger mimics $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$ faithfully, so what we need to check is that the values returned to the impersonator via the signing oracle are the same as those it would get from prover instances in

Adversary $F^{\mathcal{DS}(\mathsf{sign},sk,\cdot)}(pk)$ — Forger given signing oracle

Initialization:

(1) Choose $R_V = \text{CH}_V$ of length $vcl(k)$ at random ; $C_V \leftarrow 0$ // Coins and message counter for verifier //

(2) $p \leftarrow 0$ // Number of active prover instances //

Execute adversary I on input pk and reply to its oracle queries as follows:

- When I makes query WakeNewProver // Activate a new prover instance //
  (1) $p \leftarrow p+1$; $R_p \leftarrow \varepsilon$; `Return` p
- When I makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j})$ with $0 \le 2j < 3$ and $1 \le i \le p$
  (1) `If` $C_V \neq 0$ `then Return` $\perp$
  (2) `If` $2j = 0$ `then Return` start
  (3) `If` $2j = 2$ `then` // $\text{MSG}_1 = \mathsf{start}$ and MSG_2 is verifier challenge //
  - `If` $|\text{MSG}_2| \neq vcl(k)$ `then Return` $\perp$
  - $\text{MSG}_3 \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{MSG}_2)$ // Invoke signing oracle //
  - `Return` MSG_3
- When I makes query $\mathsf{Send}(\mathsf{vfmsg}, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1})$ with $1 \le 2j-1 \le 3$
  (1) $C_V \leftarrow C_V + 2$
  (2) `If` $2j < C_V$ `then Return` $\perp$ // Not allowed to reset the verifier //
  (3) `If` $2j-1 = 1$ `then` $\text{MSG}_2 \leftarrow \text{CH}_V$; `Return` MSG_2
  (4) `If` $2j-1 = 3$ `then`
  - $\text{SIG} \leftarrow \text{MSG}_3$
  - $\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{vf}, pk, \text{CH}_V, \text{SIG})$
  - `Return` $\varepsilon \| \mathsf{decision}$

Forgery: `Return` $(\text{CH}_V, \text{SIG})$ // Output of the forger //

Figure 13: Forger F attacking $\mathcal{DS}$, using as subroutine an impersonator I attacking the signature based ID protocol $\mathcal{ID}$ of Figure 4.

Experiment$^{\text{id-cr}}_{\mathcal{ID},I}(k)$, even in the presence of resets. This is true because the signing algorithm is stateless and deterministic. (Had the signing algorithm been probabilistic or stateful, the signature returned by a prover instance after a reset would not be obtainable via the signing oracle since the latter uses fresh coins each time or updates its state in the normal way while the reset prover instance would reuse signing coins or state.) This claim about the quality of the simulation is used to erase the distinction between the experiments in the relevant probabilities below.

Let GuessChall be the event that I makes a query $\mathsf{Send}(\mathsf{prvmsg}, p, \mathsf{start} \| \text{MSG}_2)$ in which $\text{MSG}_2 = \text{CH}_V$ equals the random challenge $R_V = \text{CH}_V$ chosen for the verifier in the initialization phase. As long as this event does not occur, F does not invoke its signing oracle on its output message CH_V, and thus, as per Definition B.1, wins if $\mathcal{DS}(\mathsf{vf}, pk, \text{CH}_V, \text{SIG}) = \mathsf{accept}$. We now bound the advantage of I as follows:

$$\begin{aligned}
\Pr[\,\text{Win}_I = \mathsf{true}\,] &= \Pr\left[\,\text{Win}_I = \mathsf{true} \wedge \overline{\text{GuessChall}}\,\right] + \Pr[\,\text{Win}_I = \mathsf{true} \wedge \text{GuessChall}\,] \\
&= \Pr[\,\text{Win}_F = \mathsf{true}\,] + \Pr[\,\text{Win}_I = \mathsf{true} \wedge \text{GuessChall}\,] \\
&\le \Pr[\,\text{Win}_F = \mathsf{true}\,] + \Pr[\,\text{GuessChall}\,] \ .
\end{aligned}$$

Adversary $E^{\mathcal{AE}(\mathsf{enc},pk,\mathsf{LR}(\cdot,\cdot,\mathrm{CB})),\mathcal{AE}(\mathsf{dec},pk,sk,\cdot)}(pk)$ — Eavesdropper given lr-encryption oracle and decryption oracle

Initialization:
(1) $C_V \leftarrow 0$ // Message counter for verifier, but no coins. //
(2) $p \leftarrow 0$ // Number of active prover instances //

Execute adversary I on input pk and reply to its oracle queries as follows:

- When I makes query $\mathsf{WakeNewProver}$ // Activate a new prover instance //
  (1) $p \leftarrow p+1$; Pick a tape R_p at random ; `Return` p
- When I makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_{2j})$ with $0 \le 2j < 3$ and $1 \le i \le p$
  (1) `If` $C_V \neq 0$ `then Return` $\perp$
  (2) `If` $2j = 0$ `then Return` start
  (3) `If` $2j = 2$ `then` // $\mathrm{MSG}_1 = \mathsf{start}$ and MSG_2 is ciphertext //
  - $\mathrm{MSG}_3 \leftarrow \mathcal{AE}(\mathsf{dec}, sk, \mathrm{MSG}_2)$ // Invoke decryption oracle //
  - `If` $|\mathrm{MSG}_3| \neq vcl(k)$ `then Return` $\perp$ `else Return` MSG_3
- When I makes query $\mathsf{Send}(\mathsf{vfmsg}, \mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_{2j-1})$ with $1 \le 2j-1 \le 3$
  (1) $C_V \leftarrow C_V + 2$
  (2) `If` $2j < C_V$ `then Return` $\perp$ // Not allowed to reset the verifier //
  (3) `If` $2j-1 = 1$ `then`
  - Let $\mathrm{CH}_0, \mathrm{CH}_1$ be random but *distinct* strings of length $vcl(k)$
  - $\mathrm{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \mathsf{LR}(\mathrm{CH}_0, \mathrm{CH}_1, \mathrm{CB}))$ // Invoke lr-encryption oracle on the messages $\mathrm{CH}_0, \mathrm{CH}_1$ //
  - $\mathrm{MSG}_2 \leftarrow \mathrm{CTXT}$ `Return` MSG_2
  (4) `If` $2j-1=3$ `then`
  - `If` $\mathrm{MSG}_3 = \mathrm{CH}_0$ `then` $\mathrm{GB} \leftarrow 0$
  - `else If` $\mathrm{MSG}_3 = \mathrm{CH}_1$ `then` $\mathrm{GB} \leftarrow 1$
  - `else` let GB be a random bit

  // The eavesdropper sets its guess bit and terminates. Nothing is returned to I in reply to this query since it is the last query and the eavesdropper has everything it needs anyway. //

Output: `Return` GB // Guess bit returned by eavesdropper //

Figure 14: Eavesdropper E attacking $\mathcal{AE}$, using as subroutine an impersonator I attacking the encryption based ID protocol $\mathcal{ID}$ of Figure 5.

Now note the probability of GuessChall is at most $q(k)/2^{vcl(k)}$ since we have assumed that the number of $\mathsf{Send}(\mathsf{prvmsg}, \cdot, \cdot)$ queries made by I is at most $q(k)$ and no information about R_V is provided during the simulation of $\mathsf{Send}(\mathsf{prvmsg}, \cdot, \cdot)$ queries. This yields Equation (1) as desired.

D.2 Proof of Theorem 3.4

Figure 14 describes the eavesdropping algorithm E attacking $\mathcal{AE}$. It runs I as a subroutine, itself responding to the latter's oracle queries so as to provide a "simulation" of the environment provided to I in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$. The eavesdropper is not in possession of the secret key sk which is used by prover instances but can compensate using its access to the decryption oracle. As usual our conventions on the way time-complexity is measured are important to it being the case that the

time-complexity of E is $t(k)$, the same as that of I. It remains to verify Equation (2).

We claim that the simulation is "perfect" —in the sense that from the point of view of I it is in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$— except for there being no reply made to the very last query of I, this being its third move message to the verifier. Indeed, the answers provided to $\mathsf{Send}(\mathsf{prvmsg},\cdot,\cdot,\cdot)$ queries are clearly the same in the simulation as in the real experiment due to invocation of the same decryption procedure, even though in the real experiment it is directly invoked and in the simulation it is invoked as an oracle without direct access to the underlying secret key. Now consider $\mathsf{Send}(\mathsf{vfmsg},\cdot)$ queries. Since both CH_0 and CH_1 are chosen at random, the ciphertext MSG_2 returned by the simulated verifier is formed by encrypting a random string, regardless of the value of the (unknown to E) challenge bit CB, and this is distributed like the corresponding ciphertext in the real experiment. In reply to its last query to the verifier, I would expect to receive the verifier decision. This is not provided in the simulation (indeed E does not know how to provide this since it does not know CB) but this is immaterial since E is in possession of I's guess MSG_3 at the challenge and, using this, outputs its own guess bit GB. This claim about the quality of the simulation is used to erase the distinction between the experiments in the relevant probabilities below.

Let GuessCiph be the event that I makes a query $\mathsf{Send}(\mathsf{prvmsg},p,\mathsf{start}\|\text{MSG}_2)$ in which $\text{MSG}_2 = \text{CTXT}$ equals the ciphertext that E obtained via its query to its lr-encryption oracle. As long as this event does not occur, E does not invoke its decryption oracle on any ciphertext returned by its lr-encryption oracle, and thus, as per Definition B.3, wins if GB = CB. We can lower bound the probability that E wins as follows:

$$\begin{aligned}\Pr[\,\text{WIN}_E = \mathsf{true}\,] &= \Pr\left[\,\text{GB} = \text{CB} \wedge \overline{\text{GuessCiph}}\,\right] \\ &\geq \Pr[\,\text{GB} = \text{CB}\,] - \Pr[\,\text{GuessCiph}\,]\ . \end{aligned} \tag{7}$$

On the other hand

$$\begin{aligned}\Pr[\,\text{GB} = \text{CB}\,] &= \Pr[\,\text{GB} = \text{CB} \mid \text{WIN}_I = \mathsf{true}\,]\cdot\Pr[\,\text{WIN}_I = \mathsf{true}\,] \\ &\quad + \Pr[\,\text{GB} = \text{CB} \mid \text{WIN}_I \neq \mathsf{true}\,]\cdot\Pr[\,\text{WIN}_I \neq \mathsf{true}\,] \\ &= 1\cdot\Pr[\,\text{WIN}_I = \mathsf{true}\,] + \left(\frac{1}{2} - \frac{1}{2^{vcl(k)}-1}\right)\cdot(1 - \Pr[\,\text{WIN}_I = \mathsf{true}\,]) \\ &= \frac{1}{2} - \frac{1}{2^{vcl(k)}-1} + \left(\frac{1}{2} + \frac{1}{2^{vcl(k)}-1}\right)\cdot\Pr[\,\text{WIN}_I = \mathsf{true}\,]\ . \end{aligned} \tag{8}$$

Above the $1/(2^{vcl(k)}-1)$ represents the probability that I does not correctly decrypt the challenge ciphertext but, unluckily for us, provides the plaintext $\text{CH}_{1-\text{CB}}$. From Equation (8) we get

$$\begin{aligned}\Pr[\,\text{WIN}_I = \mathsf{true}\,] &= \frac{2(2^{vcl(k)}-1)}{2^{vcl(k)}+1}\cdot\left(\Pr[\,\text{GB} = \text{CB}\,] - \frac{1}{2} + \frac{1}{2^{vcl(k)}-1}\right) \\ &\leq 2\cdot\Pr[\,\text{GB} = \text{CB}\,] - 1 + \frac{2}{2^{vcl(k)}+1}\ . \end{aligned}$$

Using Equation (7) and the definition of the advantage from Definition B.3 we get

$$\begin{aligned}\Pr[\,\text{WIN}_I = \mathsf{true}\,] &\leq 2\cdot(\Pr[\,\text{WIN}_E = \mathsf{true}\,] + \Pr[\,\text{GuessCiph}\,]) - 1 + \frac{2}{2^{vcl(k)}+1} \\ &= \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + 2\cdot\Pr[\,\text{GuessCiph}\,] + \frac{2}{2^{vcl(k)}+1} \\ &\leq \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + 2\cdot\frac{q(k)}{2^{vcl(k)}} + \frac{2}{2^{vcl(k)}} \\ &= \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + \frac{2q(k)+2}{2^{vcl(k)}}\ . \end{aligned}$$

Adversary $I_{\mathrm{CID}}(pk_{\mathrm{CID}})$ — Non-resetting CR1 attacker for $\mathcal{CID}$

Initialization:
(1) $C_V \leftarrow 0$ // Message counter for verifier //
(2) pick random string R_{COM} // for assimilated trapdoor commitment //
(3) $p \leftarrow 0$ // Number of active prover instances //
(4) $(pk_{\mathrm{TDC}}, sk_{\mathrm{TDC}}) \leftarrow \mathcal{TDC}(\mathsf{keygen}, k)$ // Keys for trapdoor commitment scheme //

Execute adversary I on input $pk = (pk_{\mathrm{CID}}, pk_{\mathrm{TDC}})$ and reply to its oracle queries as follows:

- When I makes query WakeNewProver // Activate a new prover instance //
  (1) $p \leftarrow p+1$; Pick a tape R_p at random ; Return p
- When I makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_{2j})$ with $0 \le 2j < 5$ and $1 \le i \le p$
  (1) If $C_V \neq 0$ then Return $\perp$
  (2) If $2j = 0$ then Return start
  (3) If $2j = 2$ then // $\mathrm{MSG}_1 = \mathsf{start}$ and MSG_2 is trapdoor commitment //
  – $\mathrm{MSG}_3 \leftarrow \mathcal{CID}(\mathsf{cmt}, sk_{\mathrm{CID}}, \epsilon; R_i)$ // Fetch commitment of CID-prover //
  – Return MSG_3
  (4) If $2j = 4$ then // MSG_4 is opening of trapdoor commitment MSG_2 //
  – parse MSG_4 as $c \| R$
  – If $\mathcal{TDC}(\mathsf{vf}, pk_{\mathrm{TDC}}, \mathrm{MSG}_2, \mathrm{MSG}_4) = \mathsf{accept}$
  – then $\mathrm{MSG}_5 \leftarrow \mathcal{CID}(\mathsf{resp}, sk_{\mathrm{CID}}, \mathrm{MSG}_3 \| c; R_i)$ // Get response from CID-prover //
  – else $\mathrm{MSG}_5 \leftarrow \perp$
  – Return MSG_5
- When I makes query $\mathsf{Send}(\mathsf{vfmsg}, \mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_{2j-1})$ with $1 \le 2j-1 \le 5$
  (1) $C_V \leftarrow C_V + 2$
  (2) If $2j < C_V$ then Return $\perp$ // Not allowed to reset the verifier //
  (3) If $2j-1 = 1$ then // Start: compute dummy trapdoor commitment for $0^{vcl(k)}$ //
  – $\mathrm{MSG}_2 \leftarrow \mathcal{TDC}(\mathsf{cmt}, pk_{\mathrm{TDC}}, 0^{vcl(k)}; R_{\mathrm{COM}})$
  – Return MSG_2
  (4) If $2j-1 = 3$ then // Forward COM, get verifier's challenge and adapt dummy commitment //
  – $\mathrm{CH}_V \leftarrow \mathcal{CID}(\mathsf{chall}, pk_{\mathrm{CID}}, \mathrm{MSG}_3)$
  – $\mathrm{MSG}_4 \leftarrow \mathcal{TDC}(\mathsf{fake}, sk_{\mathrm{TDC}}, 0^{vcl(k)} \| R_{\mathrm{COM}}, \mathrm{CH}_V)$
  – Return MSG_4
  (5) If $2j-1=5$ then // Adversary and we finish execution with verifier //
  – forward $\mathrm{MSG}_3 \| \mathrm{CH}_V \| \mathrm{MSG}_5$ to verifier in $\mathcal{CID}$

Figure 15: Impersonater I_{CID} attacking $\mathcal{CID}$ in a non-resetting CR1 model, using as subroutine an impersonator I attacking the protocol $\mathcal{ID}$ of Figure 6 in the CR1 setting.

Above we upper bounded $\Pr[\,\textsc{GuessCiph}\,]$ by the probability of guessing the underlying plaintext, using the fact that decryption is assumed unique (meaning ciphertexts of distinct plaintexts are always distinct). This yields Equation (2) as desired.

D.3 Proof of Theorem 3.6

Figure 15 shows the adversary attacking the CID-identification protocol in the non-resetting CR1 model. This algorithm gets pk_{CID} as input and tries to pass the verifier's examination by running the adversary I for $\mathcal{ID}$ as a subroutine.

Algorithm I_{CID} basically simulates the CR1-adversary I with the CID-protocol by assimilating all additional steps of $\mathcal{ID}$. Specifically, I_{CID} generates a random key pair $(pk_{\text{TDC}}, sk_{\text{TDC}})$ of the trapdoor commitment scheme and starts the simulation of I on pk_{CID} and pk_{TDC}. If this algorithm I commits to some TDCOM in some instance with the prover then I_{CID} calls the prover in $\mathcal{CID}$ to obtain COM and passes this commitment on to I. If I opens a commitment TDCOM then I_{CID} checks the correctness; if the opening is valid then forward the challenge to the prover and hand the answer to I. If the decommitment is incorrect then return $\perp$ to I without involving the prover. For a correct decommitment I_{CID} fetches the prover's response for the challenge and hands it to I.

When I finishes the phase with the prover and starts an execution with the verifier, I_{CID} commits to a dummy value $0^{vcl(k)}$. Then I sends a commitment to the verifier which I_{CID} passes to the verifier in $\mathcal{CID}$ to obtain a challenge CH_V from the verifier. Exploiting the trapdoor property and knowledge of sk_{TDC}, adversary I_{CID} finds an appropriate opening for this challenge CH_V for the dummy commitment. Note that this decommitment is identically distributed as if I_{CID} would have committed to CH_V right away. I_{CID} gives this decommitment to I and returns the answer to the verifier in $\mathcal{CID}$.

In contrast to the prover in protocol $\mathcal{ID}$ the prover in $\mathcal{CID}$ uses random coins instead of a pseudorandom function. The first step is to verify that pseudorandom values $R_i \leftarrow \mathcal{PRF}(\mathsf{eval}, \kappa, \text{TDCOM})$ instead of truly random R_i do not help I too much. To this end, we recall the hybrid model of [11] in which we replace the pseudorandom function by a random one. Namely, given protocol $\mathcal{ID}$ in the CR1 setting we denote by $\mathcal{ID}^{\text{RAND}}$ the identification scheme in which each prover instance, instead of applying a pseudorandom function to TDCOM, evaluates a random function on this value, where an independent function is selected for each prover incarnation. Although random functions are not efficiently computable, they can be simulated by assigning each new argument an independent random string, and by repeating previously given answers for the same queries. The next claim relates the advantage the adversary I might gain in $\mathcal{ID}$ compared to $\mathcal{ID}^{\text{RAND}}$ to the pseudorandomness of $\mathcal{PRF}$:

Claim D.1 Let $\mathcal{ID}$ be the identification protocol in Figure 6 and let $vcl(\cdot)$ a polynomially-bounded function. Also, let $\mathcal{PRF}$ be a pseudorandom function family. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the CR1 setting then

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) \leq q(k) \cdot \mathbf{Adv}^{\mathcal{PRF}}_{(t,q)}(k) + \mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID}^{\text{RAND}},I}(k) \; . \tag{9}$$

Proof: Given an adversary I we construct a distinguisher D for the pseudorandom function ensemble $\mathcal{PRF}$ as follows. D essentially plays the role of the honest parties, i.e., the prover and verifier, but is given oracle access to a sequence of functions $f_1, \ldots, f_{q(k)}$ which are either pseudorandom or truly random. D generates a random key pair $(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$ and starts to emulate the attack. This is done by performing all steps of the prover's incarnations and the verifier as defined by the protocol, except for the step where some prover instance i is supposed to compute $R_i \leftarrow \mathcal{PRF}(\mathsf{eval}, \kappa, \text{TDCOM})$. Instead, algorithm D replies by querying oracle f_i about TDCOM and continuing this prover's simulation for random tape R_i. The distinguisher outputs 1 if and only if the adversary is successful.

Clearly, if $f_1, \ldots, f_{q(k)}$ is a sequence of pseudorandom functions then D outputs 1 exactly if the adversary breaks $\mathcal{ID}$. On the other hand, if the functions are truly random then D outputs 1 if and

only if the adversary breaks $\mathcal{ID}^{\text{RAND}}$. The running time of D is bounded by $t(k)$ and the number of queries is at most $q(k)$. An hybrid argument now shows that this yields an algorithm distinguishing a single pseudorandom function from $\mathcal{PRF}$ and a random one; the distinguishing advantage drops by a factor $q(k)$ at most (see [2]). ▮

Hence, if the adversary I never queries a prover copy for the same prefix twice, the hybrid scheme corresponds to the setting where each prover incarnation uses an independent random tape, like the prover instances in $\mathcal{CID}$. Because such double queries can be easily eliminated by table-lookup techniques, we assume in the sequel for simplicity that I never sends the same message to the same prover instance twice.

Next we bound the probability that I finds distinct openings to a commitment TDCOM sent to the prover in $\mathcal{ID}^{\text{RAND}}$ by the maximal probability $\mathbf{Adv}_t^{\mathcal{TDC}}(k)$ of an algorithm finding a commitment with ambiguous decommitments and running in time $t(k)$. If this does not happen then I virtually mounts a non-resetting CR1 attack on $\mathcal{ID}^{\text{RAND}}$, and therefore I_{CID} a corresponing attack on $\mathcal{CID}$.

Claim D.2 If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}^{\text{RAND}}$ in the CR1 setting then for I_{CID} attacking $\mathcal{CID}$ as defined in Figure 15 we have

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID}^{\text{RAND}},I}(k) \leq \mathbf{Adv}_t^{\mathcal{TDC}}(k) + \mathbf{Adv}^{\text{id-nr-cr1}}_{\mathcal{CID},I_{\text{CID}}}(k) \; . \tag{10}$$

Proof: Conditioning on the event UNAMBIGUITY that the impersonator I does not send TDCOM with two valid decommitments to some prover incarnation, it is clear that I runs a non-resetting CR1 attack only. In this case, adversary I_{CID} wins whenever I wins. It therefore suffices to bound the probability of event $\overline{\text{UNAMBIGUITY}}$.

We claim that $\Pr\left[\overline{\text{UNAMBIGUITY}}\right]$ is at most $\mathbf{Adv}_t^{\mathcal{TDC}}(k)$. This can be seen as follows. Given a public key pk_{TDC} of the trapdoor commitment scheme we choose a pair $(pk_{\text{CID}}, sk_{\text{CID}})$ for the identification protocol and run an attack of I on $\mathcal{ID}^{\text{RAND}}$ by impersonating the honest prover and verifier. If I outputs a commitment TDCOM with distinct openings with respect to pk_{TDC} then we output this commitment with the openings, too. Apparently, the probability that we find such ambiguous decommitments equals the probability $\Pr\left[\overline{\text{UNAMBIGUITY}}\right]$, and the running time of our algorithm is bounded by $t(k)$. This completes the proof. ▮

Collecting the probabilities from Claims D.1 and D.2 yields the theorem.