Preliminary draft.

Identification Protocols Secure Against Reset Attacks

Mihir Bellare[*] Shafi Goldwasser[†] Silvio Micali[‡]

April 28, 2000

Abstract

We introduce new ID schemes that are secure even when (1) an adversary can reset the internal state and/or randomization source of the user identifying itself, and (2) when executed in an asynchronous environment like the Internet. Our new protocols and schemes are ideally suited for use by devices (like smart cards) that canot keep reliably their internal states 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.

[†] 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 3

1.1 Resettability . . . 3

1.2 Notions of security . . . 4

1.3 Three Paradigms for Achieving Secure Resettable identification . . . 5

1.4 Comparison with existing work . . . 6

2 Security of identification protocols 6

3 Primitives used and their security 12

3.1 Stateless digital signature schemes . . . 12

3.2 CCA2-secure Encryption schemes . . . 14

4 CR1-secure Identification protocols 15

4.1 A signature based protocol . . . 15

4.2 An encryption based protocol . . . 17

4.3 A zero-knowledge based protocol . . . 18

5 CR2-secure Identification protocols 19

5.1 A signature based protocol . . . 19

5.2 An encryption based protocol . . . 21

References 22

A Proofs 24

A.1 Proof of Theorem 4.1 . . . 24

A.2 Proof of Theorem 4.4 . . . 25

1 Introduction

The introduction of "zero-knowledge interactive proofs" [GMRa] was shortly followed by its best known application: smartcard based identification [FiSh]. Whereas in a zero-knowledge interactive proof a prover's goal is to prove membership in a language, in an identification protocol a prover's goal is to prove who he/she is. In [FFS] the notion of "zero knowledge proofs of knowledge" was developed and the following paradigm for identification was suggested: Let L be a hard language in NP. Each user Alice is to be identified with an $x \in L$ for which she knows a short "witness" w that $x \in L$. (Naturally, L should posses enough structure that it is possible to choose pairs (x, w) with respect to which the problem remains intractable). Then, to identify herself Alice would prove to the system manager (verifier) that she "knows" w using a zero-knowledge proof of knowledge. In particular, [FiSh] explicitly proposes to apply the above paradigm to the "quadratic residuosity" language as follows.

Example: A user A may be associated with a composite integer n and a square $x \in Z_n^*$ for which he knows a square root mod n. (E.g, A may secretly choose y and set $x = y^2 \bmod n$.) Then, when he wants to be identified (as in a remote log in), A gives a zero-knowledge proof that she knows a square root y of x as follows [GMRa]: prover A sends a random r and returns $w = r^2$ mod n to the verifier; the verifier flips a coin b and sends it to A; A returns $z = ry^b = \sqrt{wx^b} \bmod n$. If $z^2 = wx^b \bmod n$ then the verifier accepts else it rejects. By the zero-knowledge property, an enemy who listens (or even is the verifier itself) cannot falsely identify itself as A at a later time.

The protocol of [FiSh] is a more efficient variant of this basic protocol which appeared in [GMRa] to prove membership in the quadratic residuosity language. Other identity protocols were proposed subsequently (e.g. [GQ,Sch]), following the same paradigm, each focusing on better and better round/message complexity. This paradigm was especially promoted as useful for smart-card applications. Namely, a smart-card holding the witness w, would be able to convince a prover of its identity over the network by utilizing a zero knowledge proof of knowledge of w.

1.1 Resettability

A interesting question raised by [CGGM] is what happens to the above paradigm if an adversary is able to run several identification protocols (in the role of a verifier) with the same prover, each time being able to *reset the prover to the same internal state including the same random tape.* Is the above paradigm still secure, or can now the verifier adversary learn enough to impersonate the prover later on?

The answer is that any zero-knowledge proof of knowledge (and thus all proofs of identity à la Fiat-Shamir [FiSh]), are *by definition* completely breakable in the resttable model. 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 is allowed to run the protocol with the prover rewinding it a polynomial number of times (i.e resetting the prover to initial conditions with the same coins a polynomial number of times). We illustrate this for the example above.

Example Revisited: Assume that in the example above the prover is forced to execute twice with the same coins r the basic protocol. Then, by sending $b = 0$ in the first execution and $b = 1$ in the second execution, the verifier learns both r and xr and thus trivially extracts x, a square root of y mod n. Notice that the enemy needs not even be a verifier, it suffices that he has access to the smartcard and is able to reset this twice to the same state, feed it messages, and get the replies.

Is this a real security concern? We believe it is. Resetting or restoring the computational state of a device is particularly simple in 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 zero-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 spot", thereby forcing it to essentially reset itself.

Resettable zero-knowledge. The new concept of "resettable zero-knowledge interactive proofs " has been proposed by [CGGM] to address the resettability attack during a zero knowledge protocol. A protocol was defined to be "resettable zero knowledge" if it remains zero knowledge even when the verifier can run the protocol a polynomial number of times, each time resetting the prover to the same initial conditions and the same random coins. It is shown in [CGGM] how to prove *membership in any* NP *language* in resettable zero-knowledge. These results are however proofs of membership and not proofs of knowledge. (As indicated above, proofs of knowledge are by definition impossible in a resettable setting.) Thus they do not suffice to implement the Fiat-Shamir paradigm of proving identity by proving knowledge of a witness in the resettable setting. In [CGGM] a pointer to future work (this paper) promised to develop alternative paradigms for identity protocols in the resettable setting. This is one of the contributions of this work.

1.2 Notions of security

First we give a definition of security for identity protocols in a resettable setting. 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 concurrently run many identity protocols with 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). We will say that an ID scheme is secure in the CR1 setting if the probability that Vicky succeeds is negligible.

In a CR2 attack, Vicky, *while* trying to impersonate Alice (i.e attempting to identify herself as Alice to Bob the verifier), may concurrently run many identity protocols (in the verifier's role) with the real prover Alice, resetting Alice to initial conditions and interleaving executions. We will say that an ID scheme is secure in the CR2 setting if the probability that Vicky succeeds is negligible. Clearly, a CR1 attack is a special case of a CR2 attack.

Session Id's to Avoid Woman-in-the-middle. A definition of security 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 exclude it. We address them based on definitional ideas from [BR, BPR], specifically by assigning session-id's to each completed execution of an ID protocol, which the prover must generate and the verifier accept at the completion of the execution. (The idea of session ids actually goes back to Bellare, Petrank, Rackoff and Rogaway, 1996.) A successful on-line attack of Vicky on an ID scheme is defined as one having a session-id which is different from all past session-id's. As in the woman-in-the-midde attack, if the same session-id will be generated between Vicky and Bob as the one between Alice and Bob, it will not qualify as a successful on-line attack. We argue that introducing session-id's makes complete sense in practically any use of identification protocol as identification is almost always a prelude to secure sessions.

An analogy. An analogy with chosen-ciphertext attacks in encryption is useful to understand the distinction between the CR1 and CR2 settings, and helps explain why we consider the line we have drawn between them to be natural. View the decryption oracle provided to an adversary attacking an encryption scheme as the analogue of the access to prover instances given to the adversary above. View the response that the encryption adversary must provide to the challenge ciphertext as the analogue of the interaction that the adversary above must have with the verifier. Then the IND-CCA1 setting of Naor and Yung [NY2], allowing access to the decryption oracle only prior to the appearance of the challenge, is the analogue of our CR1 setting; the IND-CCA2 setting of Rackoff and Simon [RS] allowing access to the decryption oracle even after the appearance of the challenge, is the analogue of our CR2 setting. In IND-CCA2 setting the adversary can trivially win by querying its decryption oracle with the challenge ciphertext; the analogue is that our adversary could make the verifier accept by playing "man in the middle". The definitional "fix" of the IND-CCA2 setting is to simply disallow this one query. The fix in the identification setting is less trivial; it is via the use of session ids discussed above.

1.3 Three Paradigms for Achieving Secure Resettable identification

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 return to the zero-knowledge ideas and provide a new paradigm based on zero-knowledge proofs of membership (as opposed to proofs of knowledge).

The basic idea of the signature based paradigm is that 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 [FFS] as long as the signature scheme is secure in the sense of [GMRi], and also known to be secure in the concurrent non-reset setting [BCK]. 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 under this condition 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 both resettable identification schemes proven secure under minimal assumptions for both the CR1 and the CR2 settings, and also obtain some efficient specific protocols.

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 [DDN1, DDN2, BCK, DNS] 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 settings needs to be modified before it will work in the CR2 setting.

In the zero-knowledge proofs of membership paradigm, Alice convinces Bob she is Alice, by being "able to" prove membership in a hard language 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 [CGGM] . 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 [CGGM] 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 or witness-hiding proofs.

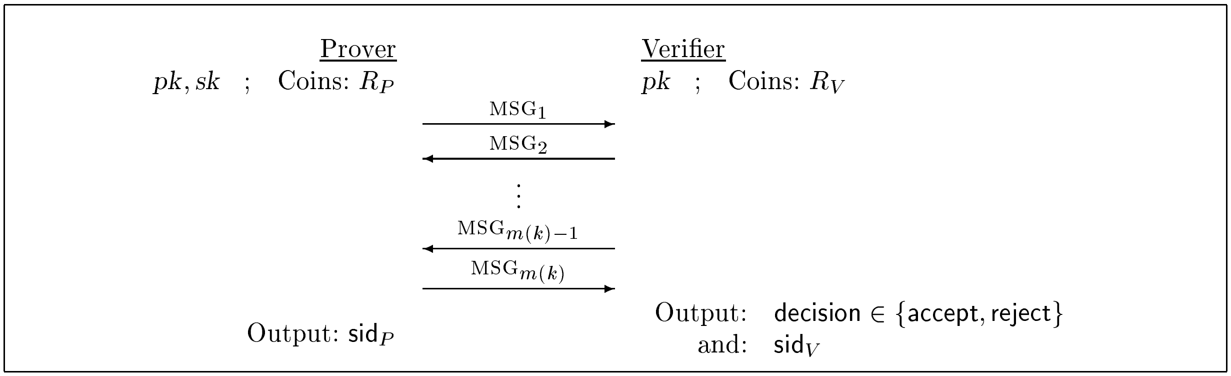
1.4 Comparison with existing work

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

In the setting of [FFS], the adversary first gets to interact with prover instances, but one by one, meaning serially, and without reset; then in a second phase, it tries to convince the verifier to accept. Our CR1 setting is an extension of this in which the adversary's capabilities are enhanced in two ways: it can query the prover instances concurrently, and it can reset them. In the setting of [BR] and succeeding works, concurrency is already allowed, both in querying prover instances and in being allowed to access these concurrently with the verifier, but reset is not considered. Our CR2 setting is thus an extension of their setting to allow reset attacks. However we also somewhat simplify their framework by restricting ourselves to identification, not identification combined with session key distribution, and to unilateral identification as opposed to mutual.

2 Security of identification protocols

The adversary model here —allowing reset attacks in a concurrent execution setting— is the strongest model 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 [FFS]— 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 [BR]— 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



$(pk, sk) \leftarrow \mathcal{ID}(\mathsf{keygen}, k)$ — 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.

definitions for both settings by appropriately adapting and extending [FFS] for the CR1 case and [BR, BPR] for the CR2 case.

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.

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.

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
    $\text{MSG}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j-1}; R_V)$
    $\text{MSG}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \text{MSG}_1 \| \cdots \| \text{MSG}_{2j}; R_P)$
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 one. (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 are two experiments. $\mathbf{Experiment}^{\text{id-sr}}_{\mathcal{ID},I}(k)$, depicted in Figure 2, 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 experiments give 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.

Each experiment 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 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

Experiment$^{\text{id-sr}}_{\mathcal{ID},I}(k)$ — 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 $\mathsf{Send}(\mathsf{prvmsg}, i, \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j})$ with $0 \le 2j < m(k)$ and $1 \le i \le p$
    (1) If $C_V \ne 0$ then Return $\perp$   // Interaction with prover instance allowed only before interaction with verifier begins //
    (2) $\textsc{msg}_{2j+1} \leftarrow \mathcal{ID}(\mathsf{prvmsg}, sk, \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j}; R_i)$
    (3) Return $\textsc{msg}_{2j+1}$
• When $I$ makes query $\mathsf{Send}(\mathsf{vfmsg}, \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j-1})$ 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 $\textsc{msg}_{2j} \leftarrow \mathcal{ID}(\mathsf{vfmsg}, pk, \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j-1}; R_V)$ ; Return $\textsc{msg}_{2j}$
    (4) If $2j-1 = m(k)$ then decision $\leftarrow \mathcal{ID}(\mathsf{vfend}, pk, \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j}; R_V)$
    (5) Return decision

Did I win? When $I$ has terminated set $\textsc{Win}_I$ = true if decision = accept.
```

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

prefixes to be queried. (For example the adversary might try $\textsc{msg}_1 \| \textsc{msg}_2$ for many different values of $\textsc{msg}_2$, 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 $x = \textsc{msg}_1 \| \cdots \| \textsc{msg}_{2j-1}$ 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:

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**$^{\text{id-sr}}_{\mathcal{ID},I}(k)$

```
$\textbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$ — 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 $\mathsf{WakeNewProver}$   // 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 $\text{MSG}_{2j}$
    (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.

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 $\textbf{Experiment}^{\text{id-sr}}_{\mathcal{ID},I}(k)$. 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 [BR], is based on the idea of "matching conversations" and corresponds to a very stringent security requirement. Another possible definition is that of [BPR] 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 $\text{W}\textsc{in}_I$ is set accordingly in $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$. Session ids are public in the sense that the adversary gets to see those created by any instances with which it interacts.

Definitions 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

$$\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID},I}(k) = \Pr\left[\, \text{W}\textsc{in}_I = \mathsf{true} \,\right]$$

where the probability is with respect to $\mathbf{Experiment}^{\text{id-sr}}_{\mathcal{ID},I}(k)$. Protocol $\mathcal{ID}$ is said to be *polynomially-secure in the* CR1 *setting* if $\mathbf{Adv}^{\text{id-cr1}}_{\mathcal{ID}}(\cdot)$ 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 $\mathbf{Experiment}^{\text{id-sr}}_{\mathcal{ID},I}(k)$, including the time taken for initialization, computation of replies to adversary oracle queries, and computation of $\text{W}\textsc{in}_I$. 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 $\mathbf{Experiment}^{\text{id-sr}}_{\mathcal{ID},I}(k)$. 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.

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

$$\mathbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) = \Pr\left[\, \text{W}\textsc{in}_I = \mathsf{true} \,\right]$$

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

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

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

Figure 4: The *digital signature scheme description* $\mathcal{DS}$ describes all functionalities associated to the signature scheme.

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 [BR, BPR] 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.

3 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.

3.1 Stateless digital signature schemes

Signature schemes. A digital signature scheme is specified by a description function $\mathcal{DS}$, which, as indicated in Figure 4, 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 [GMRi].

Definition 3.1 [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

```
$\mathbf{Experiment}^{\mathrm{ds}}_{\mathcal{DS},F}(k)$
$(pk, sk) \leftarrow \mathcal{DS}(\mathsf{keygen}, k)$ ; $\textsc{Win}_F \leftarrow \mathsf{false}$
$(\textsc{msg}, \textsc{sig}) \leftarrow F^{\mathcal{DS}(\mathsf{sign}, sk, \cdot)}(pk)$
If $\mathcal{DS}(\mathsf{verify}, pk, \textsc{msg}, \textsc{sig}) = \mathsf{accept}$ and $F$ never made oracle query $\textsc{msg}$
    then $\textsc{Win}_F \leftarrow \mathsf{true}$
```

The *advantage* of forger F is

$$\mathbf{Adv}^{\mathrm{ds}}_{\mathcal{DS},F}(k) = \Pr\left[\, \textsc{Win}_F = \mathsf{true} \,\right]$$

where the probability is with respect to $\mathbf{Experiment}^{\mathrm{ds}}_{\mathcal{DS},F}(k)$. Digital signature scheme $\mathcal{DS}$ is said to be *polynomially-secure* if $\mathbf{Adv}^{\mathrm{ds}}_{\mathcal{DS}}(\cdot)$ 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 $\mathbf{Experiment}^{\mathrm{ds}}_{\mathcal{DS},F}(k)$, 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 [GMRi]— 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 3.1 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 3.1 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 [GMRi, BeMi, NY1, Ro] are (randomized and) stateful. The more efficient schemes of [DwNa, CD] are also (randomized and) stateful. Examples of (randomized but) stateless schemes are those of [GHR, CS2]. 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 [GMRi] 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 K specifying an instance F_K of a family of pseudorandom functions [GGM], and to sign

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

Figure 5: The *asymmetric encryption scheme description* $\mathcal{AE}$ describes all functionalities associated to the encryption scheme.

message $\textsc{msg}$ compute $R_{\mathrm{MSG}} = F_K(\textsc{msg})$ and use R_{MSG} as the coins for the signing algorithm. Combining this with Rompel's result [Ro] implies:

Proposition 3.2 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 [GHR, CS2]. 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 [DwNa, CD] but they are less efficient than those of [GHR, CS2] and also are stateful. Making a stateful scheme stateless seems to be more costly than derandomizing an already stateless scheme.)

3.2 CCA2-secure Encryption schemes

Encryption schemes. An asymmetric encryption scheme is specified by a description function $\mathcal{AE}$, which as indicated, in Figure 5, specifies how keys are generated, how messages are encrypted, and how ciphertexts are decrypted. (As usual it is required that if ciphertext $\textsc{ctxt}$ is generated via $\mathcal{AE}(\mathsf{enc}, pk, \textsc{msg})$ then $\mathcal{AE}(\mathsf{dec}, pk, sk, \textsc{ctxt})$ returns $\textsc{msg}$.) 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 [BBM], 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 [RS]. Define $\mathsf{LR}(\textsc{msg}_0, \textsc{msg}_1, b) = \textsc{msg}_b$ for any equal-length strings $\textsc{msg}_0, \textsc{msg}_1$ and bit b.

Definition 3.3 [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 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

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

The *advantage* of eavesdropper E is

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

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

The most efficient scheme proven to meet this notion under a standard assumption is that of [CS1].

4 CR1-secure Identification protocols

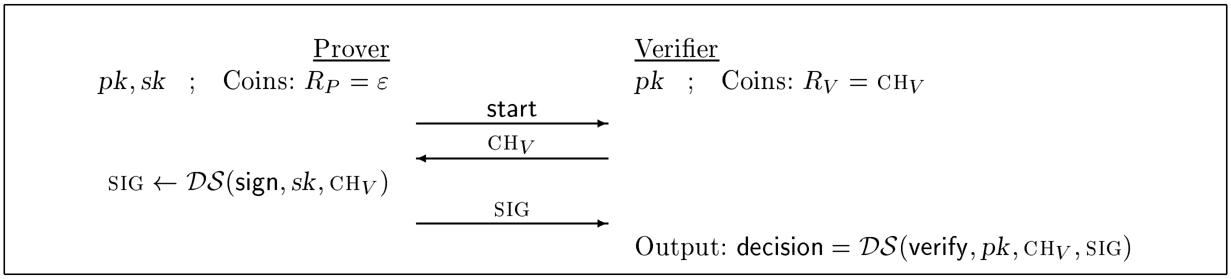
Three paradigms are illustrated: signature based, encryption based, and zero-knowledge based.

4.1 A signature based protocol

This section assumes knowledge of background in digital signatures as summarized in Section 3.1.

Signature based identification. A natural identification protocol is for the verifier to issue a random challenge $\textsc{ch}_V$ and the prover respond with a signature of $\textsc{ch}_V$ 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 [FFS] as long as the signature scheme meets the notion of security of [GMRi] provided in Definition 3.1. (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 [BCK]. (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 Section 3.1, this is not covered by the usual notion of a chosen-message attack used to define security of signature schemes in [GMRi]. 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


Prover
pk, sk ; Coins: $R_P = \varepsilon$
Verifier
pk ; Coins: $R_V = \text{CH}_V$
start
CH_V
$\text{SIG} \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{CH}_V)$
SIG
Output: $\mathsf{decision} = \mathcal{DS}(\mathsf{verify}, pk, \text{CH}_V, \text{SIG})$

$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{DS}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ has same key generation process as $\mathcal{DS}$	
$\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_P)$ where $\lvert R_P\rvert = 0$ – Parse x as $\text{MSG}_1 \Vert \cdots \Vert \text{MSG}_l$ – If $l \notin \{0, 2\}$ then Return $\bot$ – If $l = 0$ then Return start – If $\lvert \text{MSG}_2 \rvert \neq vcl(k)$ then Return $\bot$ – $\text{CH}_V \leftarrow \text{MSG}_2$; $\text{SIG} \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{CH}_V)$ – Return SIG	$\mathcal{ID}(\mathsf{prvsid}, sk, x; R_P)$ where $\lvert R_P\rvert = 0$ – Return ε
$\mathcal{ID}(\mathsf{vfmsg}, pk, x; R_V)$ where $\lvert R_V\rvert = vcl(k)$ – Parse x as $\text{MSG}_1 \Vert \cdots \Vert \text{MSG}_l$ – If $l \neq 1$ then Return $\bot$ – $\text{CH}_V \leftarrow R_V$ – Return CH_V	$\mathcal{ID}(\mathsf{vfend}, pk, x; R_V)$ where $\lvert R_V\rvert = vcl(k)$ – Parse x as $\text{MSG}_1 \Vert \cdots \Vert \text{MSG}_l$ – If $l \neq 3$ or $\text{MSG}_2 \neq R_V$ then Return $\bot$ – $\text{CH}_V \leftarrow \text{MSG}_2$; $\text{SIG} \leftarrow \text{MSG}_3$ – $\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{verify}, pk, \text{CH}_V, \text{SIG})$ – Return $\varepsilon \Vert \mathsf{decision}$

Figure 6: 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.

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. Section 3.1 explains 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 6 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 3.1 for the meanings of terms used in the theorem below, and to Section A.1 for the proof.

Theorem 4.1 [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 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 a forger F attacking $\mathcal{DS}$ such that

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

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 4.2 [Polynomial-security of the signature based ID scheme in the $\mathsf{CR1}$ **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 6. If $\mathcal{DS}$ is polynomially-secure then $\mathcal{ID}$ is polynomially-secure in the $\mathsf{CR1}$ setting. ▮

Corollary 4.2 together with Proposition 3.2 imply:

Corollary 4.3 [Existence of an ID scheme polynomially-secure in the $\mathsf{CR1}$ **setting]** Assume there exists a one-way function. Then there exists an identification scheme that is polynomially-secure in the $\mathsf{CR1}$ setting.

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

4.2 An encryption based protocol

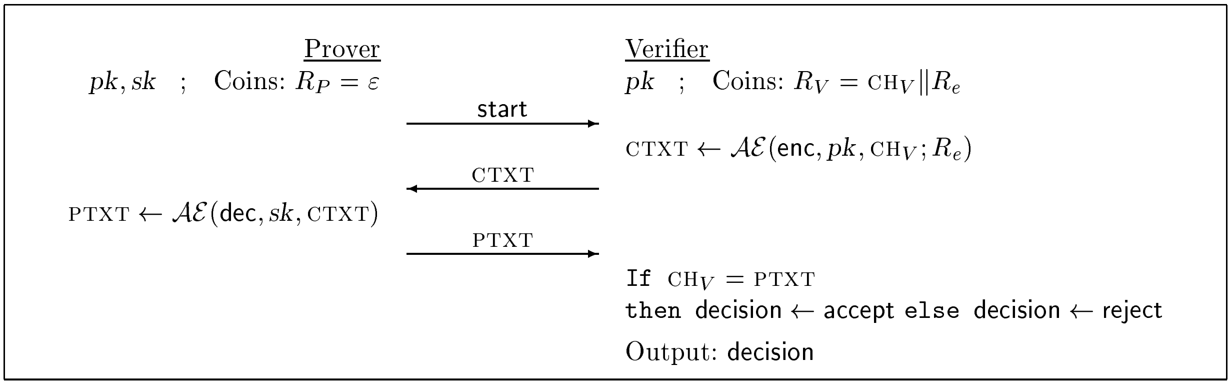
Encryption based identification. The idea is simple: the prover proves its identity by proving its ability to decrypt a ciphertext sent by the verifier. This basic idea goes back to early work in entity authentication where the encryption was usually symmetric (ie. private-key based). These early protocols however had no supporting definitions or analysis. The first "modern" treatment is that of [DDN1, DDN2] who considered the paradigm with regard to provding deniable authentication and identified non-malleability under chosen-cihphertext attack —equivalently, indistinguishability under chosen-ciphertext attack [BDPR, DDN2]— as the security property required of the encryption scheme. Results of [BCK, DNS, DDN1, DDN2] 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 7 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 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 invokation of the encryption algorithm. Refer to Definition 2.1 and Definition 3.3 for the meanings of terms used in the theorem below, and to Section A.2 for the proof.

Theorem 4.4 [Concrete security of the encryption based ID scheme in the $\mathsf{CR1}$ **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 7. If I is an adversary of time-complexity $t(\cdot)$ and query-complexity $q(\cdot)$ attacking $\mathcal{ID}$ in the $\mathsf{CR1}$ setting then there exists an eavesdropper E attacking $\mathcal{AE}$ such that

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

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.



Prover: pk, sk ; Coins: $R_P = \varepsilon$

Verifier: pk ; Coins: $R_V = \mathrm{CH}_V \| R_e$

$\xrightarrow{\mathsf{start}}$

$\mathrm{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \mathrm{CH}_V; R_e)$

$\xleftarrow{\mathrm{CTXT}}$

$\mathrm{PTXT} \leftarrow \mathcal{AE}(\mathsf{dec}, sk, \mathrm{CTXT})$

$\xrightarrow{\mathrm{PTXT}}$

If $\mathrm{CH}_V = \mathrm{PTXT}$
then decision ← accept else decision ← reject

Output: decision

$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{AE}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ has same key generation process as $\mathcal{AE}$	
$\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_P)$ where $R_P = \varepsilon$ – Parse x as $\mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_l$ – If $l \notin \{0, 2\}$ then Return $\perp$ – If $l = 0$ then Return start – $\mathrm{CTXT} \leftarrow \mathrm{MSG}_2$; $\mathrm{PTXT} \leftarrow \mathcal{AE}(\mathsf{dec}, sk, \mathrm{CTXT})$ – If $\lvert\mathrm{PTXT}\rvert \neq vcl(k)$ then Return $\perp$ – Return PTXT	$\mathcal{ID}(\mathsf{prvsid}, sk, x; R_P)$ where $R_P = \varepsilon$ – Return ε
$\mathcal{ID}(\mathsf{vfmsg}, pk, x; R_V)$ – Parse R_V as $\mathrm{CH}_V \| R_e$ with $\lvert\mathrm{CH}_V\rvert = vcl(k)$ – Parse x as $\mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_l$ – If $l \neq 1$ then Return $\perp$ – $\mathrm{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \mathrm{CH}_V; R_e)$ – Return CTXT	$\mathcal{ID}(\mathsf{vfend}, pk, x; R_V)$ – Parse R_V as $\mathrm{CH}_V \| R_e$ with $\lvert\mathrm{CH}_V\rvert = vcl(k)$ – Parse x as $\mathrm{MSG}_1 \| \cdots \| \mathrm{MSG}_l$ – If $l \neq 3$ then Return $\perp$ – $\mathrm{PTXT} \leftarrow \mathrm{MSG}_3$; sid $\leftarrow \mathrm{CH}_V$ – If $\mathrm{PTXT} = \mathrm{CH}_V$ then decision ← accept else decision ← reject – Return $\varepsilon \|$decision

Figure 7: 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.

This immediately implies the following:

Corollary 4.5 [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 7. If $\mathcal{AE}$ is polynomially-secure against chosen-ciphertext attack then $\mathcal{ID}$ is polynomially-secure in the CR1 setting. ▮

4.3 A zero-knowledge based protocol

As we discussed in the Introduction the idea of [FFS] 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.

Resetable 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 [CGGM]. In [CGGM] 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 [CGGM] 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.

5 CR2-secure Identification protocols

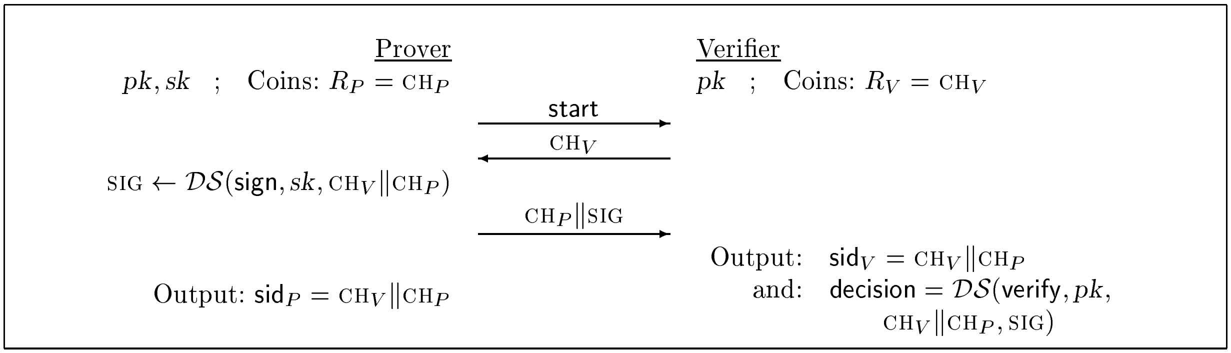
The protocols of Section 4 are not secure in the CR2 setting. We show how the same paradigms can be applied to yield modified protocols that are secure in the CR2 setting.

5.1 A signature based protocol

The signature based protocol of Figure 6 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 6 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. (This protocol is related to the signature based MT-authenticator of [BCK].) 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 8 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


Prover
pk, sk ; Coins: $R_P = \text{CH}_P$
Verifier
pk ; Coins: $R_V = \text{CH}_V$
start
CH_V
$\text{SIG} \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \text{CH}_V \| \text{CH}_P)$
$\text{CH}_P \| \text{SIG}$
Output: $\mathsf{sid}_V = \text{CH}_V \| \text{CH}_P$
and: $\mathsf{decision} = \mathcal{DS}(\mathsf{verify}, pk, \text{CH}_V \| \text{CH}_P, \text{SIG})$
Output: $\mathsf{sid}_P = \text{CH}_V \| \text{CH}_P$

$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{DS}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ has same key generation process as $\mathcal{DS}$	
$\mathcal{ID}(\mathsf{prvmsg}, sk, x; R_P)$ where $\lvert R_P \rvert = pcl(k)$ – 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 $\lvert \text{MSG}_2 \rvert \neq vcl(k)$ 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 $\lvert R_P \rvert = pcl(k)$ – Parse x as $\text{MSG}_1 \| \cdots \| \text{MSG}_l$ – If $l \neq 3$ or $\lvert \text{MSG}_2 \rvert \neq vcl(k)$ 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 $\lvert R_V \rvert = vcl(k)$ – 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 $\lvert R_V \rvert = vcl(k)$ – 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 $\lvert \text{CH}_P \rvert = pcl(k)$ – $\text{CH}_V \leftarrow \text{MSG}_2$; $\mathsf{sid}_V \leftarrow \text{CH}_V \| \text{CH}_P$ – $\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{verify}, pk, \text{CH}_V \| \text{CH}_P, \text{SIG})$ – Return $(\mathsf{sid}, \mathsf{decision})$

Figure 8: 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.

and the length $pcl(k)$ of the prover's random challenge. The random tape, for each party, is its challenge. Refer to Definition 2.2 and Definition 3.1 for the meanings of terms used in the theorem below. The proof is similar to that of Theorem 4.1 and is omitted.

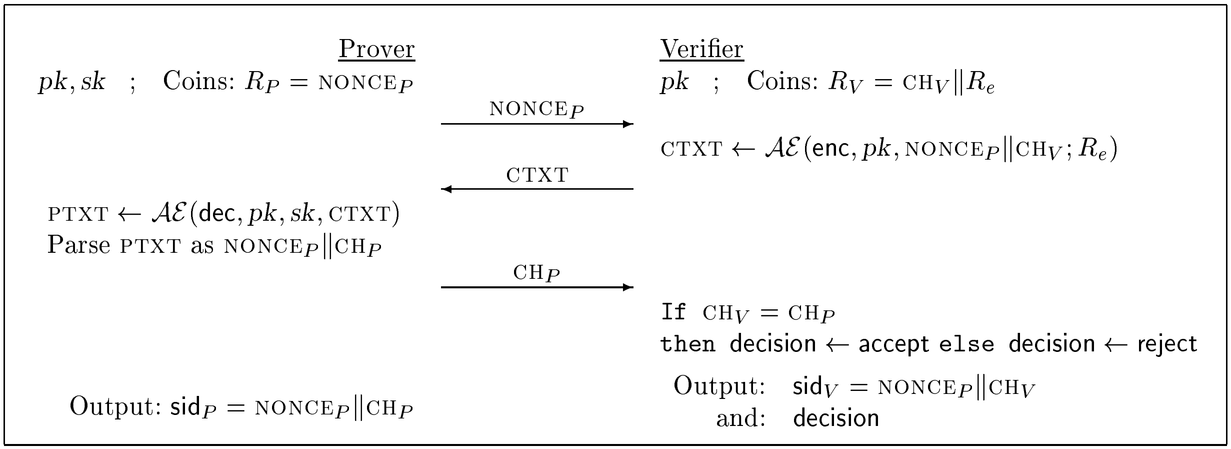
Theorem 5.1 [Concrete security of the signature based ID scheme in the CR2 setting] Let $\mathcal{DS}$ be a deterministic, stateless signature scheme, let $vcl(\cdot)$ and $pcl(\cdot)$ be polynomially-bounded functions, and let $\mathcal{ID}$ be the associated identification scheme as per Figure 8. If 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

$$\mathbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{ds}}_{\mathcal{DS},F}(k) + \frac{q(k)}{2^{vcl(k)}} + \frac{q(k)^2 - q(k)}{2^{pcl(k)+1}} . \tag{3}$$

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 5.2 [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


Prover
pk, sk ; Coins: $R_P = \text{NONCE}_P$
Verifier
pk ; Coins: $R_V = \text{CH}_V \| R_e$
NONCE_P
$\text{CTXT} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \text{NONCE}_P \| \text{CH}_V; R_e)$
CTXT
$\text{PTXT} \leftarrow \mathcal{AE}(\mathsf{dec}, pk, sk, \text{CTXT})$
Parse PTXT as $\text{NONCE}_P \| \text{CH}_P$
CH_P
If $\text{CH}_V = \text{CH}_P$
then decision ← accept else decision ← reject
Output: $\mathsf{sid}_P = \text{NONCE}_P \| \text{CH}_P$
Output: $\mathsf{sid}_V = \text{NONCE}_P \| \text{CH}_V$
and: decision

$\mathcal{ID}(\mathsf{keygen}, k) = \mathcal{AE}(\mathsf{keygen}, k)$ — $\mathcal{ID}$ 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 $\text{CH}_P$
```

```
$\mathcal{ID}(\mathsf{prvsid}, sk, x; R_P)$ 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 9: 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.

the associated identification scheme as per Figure 8. If $\mathcal{DS}$ is polynomially-secure then $\mathcal{ID}$ is polynomially-secure in the CR2 setting. ▌

Corollary 5.3 [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.

5.2 An encryption based protocol

The encryption based protocol of Figure 7 (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 $\mathrm{CH}_P \| \mathrm{PTXT}$. 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 8 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 9 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 3.3 for the meanings of terms used in the theorem below. The proof is similar to that of Theorem 4.4 and is omitted.

Theorem 5.4 [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 9. 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

$$\mathbf{Adv}^{\text{id-cr2}}_{\mathcal{ID},I}(k) \leq \mathbf{Adv}^{\text{lr-cca}}_{\mathcal{AE},E}(k) + \frac{2q(k)+2}{2^{vcl(k)}} + \frac{q(k)^2 - q(k)}{2^{pcl(k)}} \,. \qquad (4)$$

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 5.5 [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 9. If $\mathcal{AE}$ is polynomially-secure against chosen-ciphertext attack then $\mathcal{ID}$ is polynomially-secure in the CR2 setting. ▌

References

[BBM] 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. ??, B. Preneel ed., Springer-Verlag, 2000.

[BCK] 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.

[BDPR] 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.

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

[BPR] 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. ??, B. Preneel ed., Springer-Verlag, 2000.

[BR] 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.

[CGGM] 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.

[CD] 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.

[CS1] 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.

[CS2] 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.

[DDN1] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography", TR CS95-27, Weizmann Institute. Preliminary version in *Proceedings of the 23rd Annual Symposium on the Theory of Computing*, ACM, 1991.

[DDN2] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography", Manuscript, 1999.

[DwNa] 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.

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

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

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

[FiSh] 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.

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

[GMRa] 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.

[GMRi] 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.

[GHR] 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.

[NY1] 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.

[NY2] 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.

[RS] 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.

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

Initialization:
(1) Choose $R_V = \textsc{ch}_V$ 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 $\mathsf{WakeNewProver}$  // Activate a new prover instance //
    (1) $p \leftarrow p+1$ ; $R_p \leftarrow \varepsilon$ ; Return $p$
• When $I$ makes query $\mathsf{Send}(\mathsf{prvmsg}, i, \textsc{msg}_1\|\cdots\|\textsc{msg}_{2j})$ with $0 \le 2j < 3$ and $1 \le i \le p$
    (1) If $C_V \ne 0$ then Return $\perp$
    (2) If $2j = 0$ then Return start
    (3) If $2j = 2$ then  // $\textsc{msg}_1 = \mathsf{start}$ and $\textsc{msg}_2$ is verifier challenge //
        – If $|\textsc{msg}_2| \ne vcl(k)$ then Return $\perp$
        – $\textsc{msg}_3 \leftarrow \mathcal{DS}(\mathsf{sign}, sk, \textsc{msg}_2)$  // Invoke signing oracle //
        – Return $\textsc{msg}_3$
• When $I$ makes query $\mathsf{Send}(\mathsf{vfmsg}, \textsc{msg}_1\|\cdots\|\textsc{msg}_{2j-1})$ 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 $\textsc{msg}_2 \leftarrow \textsc{ch}_V$ ; Return $\textsc{msg}_2$
    (4) If $2j-1=3$ then
        – $\textsc{sig} \leftarrow \textsc{msg}_3$
        – $\mathsf{decision} \leftarrow \mathcal{DS}(\mathsf{verify}, pk, \textsc{ch}_V, \textsc{sig})$
        – Return $\varepsilon\|\mathsf{decision}$

Forgery: Return $(\textsc{ch}_V, \textsc{sig})$  // Output of the forger //
```

Figure 10: Forger F attacking $\mathcal{DS}$, using as subroutine an impersonator I attacking the signature based ID protocol $\mathcal{ID}$ of Figure 6.

[Ro] 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 Proofs

A.1 Proof of Theorem 4.1

Figure 10 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 $\mathbf{Experiment}^{\text{id-cr}}_{\mathcal{ID},I}(k)$, 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 3.1, wins if $\mathcal{DS}(\mathsf{verify}, pk, \text{CH}_V, \text{SIG}) = \mathsf{accept}$. We now bound the advantage of I as follows:

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

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.

A.2 Proof of Theorem 4.4

Figure 11 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

```
Adversary $E^{\mathcal{AE}(\mathsf{enc},pk,\mathsf{LR}(\cdot,\cdot,\textsc{cb})),\,\mathcal{AE}(\mathsf{dec},pk,sk,\cdot)}(pk)$ — 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 $\mathsf{Send}(\mathsf{prvmsg}, i, \textsc{msg}_1\|\cdots\|\textsc{msg}_{2j})$ with $0 \le 2j < 3$ and $1 \le i \le p$
    (1) If $C_V \ne 0$ then Return $\perp$
    (2) If $2j = 0$ then Return start
    (3) If $2j = 2$ then   // $\textsc{msg}_1 = \mathsf{start}$ and $\textsc{msg}_2$ is ciphertext //
        – $\textsc{msg}_3 \leftarrow \mathcal{AE}(\mathsf{dec}, sk, \textsc{msg}_2)$   // Invoke decryption oracle //
        – If $|\textsc{msg}_3| \ne \mathit{vcl}(k)$ then Return $\perp$ else Return $\textsc{msg}_3$
• When $I$ makes query $\mathsf{Send}(\mathsf{vfmsg}, \textsc{msg}_1\|\cdots\|\textsc{msg}_{2j-1})$ 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 $\textsc{ch}_0, \textsc{ch}_1$ be random but distinct strings of length $\mathit{vcl}(k)$
        – $\textsc{ctxt} \leftarrow \mathcal{AE}(\mathsf{enc}, pk, \mathsf{LR}(\textsc{ch}_0, \textsc{ch}_1, \textsc{cb}))$   // Invoke lr-encryption oracle on the messages
          $\textsc{ch}_0, \textsc{ch}_1$ //
        – $\textsc{msg}_2 \leftarrow \textsc{ctxt}$ Return $\textsc{msg}_2$
    (4) If $2j-1=3$ then
        – If $\textsc{msg}_3 = \textsc{ch}_0$ then $\textsc{gb} \leftarrow 0$
        – else If $\textsc{msg}_3 = \textsc{ch}_1$ then $\textsc{gb} \leftarrow 1$
        – else let $\textsc{gb}$ 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 $\textsc{gb}$   // Guess bit returned by eavesdropper //
```

Figure 11: Eavesdropper E attacking $\mathcal{AE}$, using as subroutine an impersonator I attacking the encryption based ID protocol $\mathcal{ID}$ of Figure 7.

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 $\mathsf{Send}(\mathsf{prvmsg}, p, \mathsf{start}\|\textsc{msg}_2)$ in which $\textsc{msg}_2 = \textsc{ctxt}$ 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 3.3, wins if $\textsc{gb} = \textsc{cb}$. We can lower bound the probability that E wins as follows:

$$\begin{aligned}\Pr\left[\,\textsc{Win}_E = \mathsf{true}\,\right] &= \Pr\left[\,\textsc{gb} = \textsc{cb} \wedge \overline{\textsc{GuessCiph}}\,\right] \\ &\ge \Pr\left[\,\textsc{gb} = \textsc{cb}\,\right] - \Pr\left[\,\textsc{GuessCiph}\,\right] \,. \end{aligned} \tag{5}$$

On the other hand

$$\begin{aligned}
\Pr[\,\mathrm{GB}=\mathrm{CB}\,] &= \Pr[\,\mathrm{GB}=\mathrm{CB} \mid \textsc{Win}_I=\mathsf{true}\,]\cdot\Pr[\,\textsc{Win}_I=\mathsf{true}\,] \\
&\qquad + \Pr[\,\mathrm{GB}=\mathrm{CB} \mid \textsc{Win}_I\neq\mathsf{true}\,]\cdot\Pr[\,\textsc{Win}_I\neq\mathsf{true}\,] \\
&= 1\cdot\Pr[\,\textsc{Win}_I=\mathsf{true}\,]+\left(\frac{1}{2}-\frac{1}{2^{vcl(k)}-1}\right)\cdot(1-\Pr[\,\textsc{Win}_I=\mathsf{true}\,]) \\
&= \frac{1}{2}-\frac{1}{2^{vcl(k)}-1}+\left(\frac{1}{2}+\frac{1}{2^{vcl(k)}-1}\right)\cdot\Pr[\,\textsc{Win}_I=\mathsf{true}\,]\ . \qquad (6)
\end{aligned}$$

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 $\mathrm{CH}_{1-\mathrm{CB}}$. From Equation (6) we get

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

Using Equation (5) and the definition of the advantage from Definition 3.3 we get

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

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.