A calculus to detect guessing attacks

Bogdan Groza and Marius Minea

Politehnica University of Timişoara and Institute e-Austria Timişoara *
bogdan.groza@aut.upt.ro marius@cs.upt.ro

Abstract. We present a calculus for detecting guessing attacks, based on oracles that instantiate cryptographic functions. Adversaries can *observe* oracles, or *control* them either on-line or off-line. These relations can be established by protocol analysis in the presence of a Dolev-Yao intruder, and the derived guessing rules can be used together with standard intruder deductions. Our rules also handle partial verifiers that fit more than one secret. We show how to derive a known weakness in the Anderson-Lomas protocol, and new vulnerabilities for a known faulty ATM system.

1 Introduction and related work

Analyzing vulnerability to guessing attacks is of high practical relevance. A value is deemed guessable if it has small entropy (is chosen from a small cardinality set), and the guess can be verified. An adversary can perform guessing by off-line computation, or on-line, exploiting the interaction with honest participants.

Conceptually, guessing involves two steps. Any protocol must have a *generation oracle* which computes some value (the *verifier*), given the secret as input. Next, a boolean *verification oracle* compares a verifier for the guess with one computed for the actual secret. We use the term oracle for an abstract object that produces a value, regardless of how the computation is done. In particular, the adversary might use other participants as on-line oracles for this purpose.

Separating the verifier generation from the verification itself, and modeling them as oracles is key to our analysis of guessing attacks in both off-line and on-line settings. It is often argued that on-line guessing can be blocked after a threshold of incorrect guesses. However, if the adversary's guesses are cached as valid protocol interactions, relying on blocking is not a justified defense.

Our analysis identifies various guessing situations with partial or complete view over inputs and outputs of oracles and with off-line, on-line or blockable on-line oracle access. We provide inference rules which can detect guessing attacks in these situations. Once such a vulnerability is detected, it is up to further review to decide if it can be removed by limiting protocol runs. We will also distinguish pre-computed dictionary guessing as a particularly dangerous case: the adversary can build an off-line dictionary which is reusable and constitutes a time-memory trade-off.

* This work is supported in part by FP7-ICT-2007-1 project 216471, AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures

Related work

A classification into off-line, detectable, and undetectable on-line guessing attacks is given in [8], recognizing that principals can be used to perform computation, without this being detectable. However, attack detection is not formalized.

Lowe's rules [11] construct new terms of intruder knowledge from the guessed value. Tracking that a term is obtained in two different ways confirms the guess. Special cases avoid false deductions. In [6], substituting the guess with a fresh term provides the second derivation; [7] has a dual set of Dolev-Yao deduction rules with an explicit comparison rule and gives complexity results. In [9], an intruder checks that two maps of terms constructed by exhaustive candidate enumeration correspond on exactly one entry. This approach can model simultaneous guesses of several message components but is limited to off-line attacks.

Equational theories and static equivalence are used in [5] for applied pi-calculus, and in [1] showing computational indistinguishability; [3] uses a constraint solving algorithm for an equational theory given as a convergent term rewriting system. Blanchet's tool ProVerif [4] detects off-line guessing attacks.

Our approach is based on Dolev-Yao-style deductions, with an adversary observing or controlling oracles for functions that are injective in the secret. Correct guessing is confirmed by checking relations on inputs and outputs of the oracle, or on the output of its inverse (e.g., in the case of encryption). We extend this to functions that match several secrets, but allow verification based on more than one observation, giving guessing a probabilistic meaning. Our guessing rules contain bounds that are sufficient to achieve a correct guess in the average case.

2 Adversary relations with oracles

We write $A \rightsquigarrow x$ if A can guess x, and $A \rightsquigarrow_D x$ for guessing using pre-computed dictionaries. These are a space-time tradeoff: if $A \rightsquigarrow_D x$, then A can also guess without pre-computed dictionaries by just repeating the dictionary construction.

A value is guessed only if it is also verified. Thus, if A can guess x, then A knows x, i.e., a guessed value can be used in further reasoning, together with the usual rules that describe how a Dolev-Yao intruder can acquire its knowledge.

We denote by $O^f(\cdot)$ an oracle which computes the value of function f for any provided input. We define two relations between adversary and oracles: *observes* and *controls*, with different variants: off-line, on-line and on-line blockable, when the adversary presence is detected and the protocol stopped.

The adversary *observes* the output of oracle $O^f(\cdot)$, written $Adv \blacktriangleleft O^f(\cdot)$, if a protocol state is reachable where Adv knows $f(T)$ for some term T. Thus, *observes* is a protocol property. The placeholders $\blacktriangleright$ and $\triangleright$ specify whether oracle inputs are known, completely or in part, e.g., $Adv \blacktriangleleft O^f(\triangleright)$ means that Adv observes the output of an oracle but knows only part of its input. This allows us to relate the adversary's observation of oracle outputs with that of its inputs.

Adv may observe $O^f(\cdot)$ using *on-line* access to the protocol, initial knowledge, standard Dolev-Yao deductions, and off-line computation. In a stronger case, Adv might know the function f, and apply it *off-line* to any known term.

Adv controls the oracle $O^f(\cdot)$, i.e., can compute $f(x)$ for an input x of its choice if for any $x \in \mathrm{dom}\, f$ chosen by *Adv* in the initial protocol state, a state where *Adv* knows $f(x)$ is reachable, regardless of the actions chosen by the other participants. As with observations, this can occur off-line or on-line.

In the strongest case, *Adv* knows all oracle parameters, and can compute f *off-line*: $Adv\ \mathrm{ctl}^{off} O^f(\cdot)$. Second, *Adv* can have *on-line* control of an oracle if a protocol participant provides this service. However, it is important to distinguish whether the protocol ends normally or adversary intervention may be detected.

Let (1) $A \to B : m$ and (2) $B \to A : E_k(m)$. *Adv* can send B any value m; the protocol ends normally, and *Adv* knows $E_k(m)$. Thus, *Adv* has unrestricted, *non-blockable* online control of the oracle, $Adv\ \mathrm{ctl}^{nb} O^f(\cdot)$. Now let (1) $A \to B : m_A$, (2) $B \to A : m_B, E_k(m_A)$, (3) $A \to B : E_k(m_B)$. Again, *Adv* obtains $E_k(m)$, but cannot compute $E_k(m_B)$, so the protocol is not completed. *Adv* has *blockable control* of the oracle, denoted $Adv\ \mathrm{ctl}^{bl} O^f(\cdot)$, since incorrect termination may be detected and subsequent protocol runs be blocked by the honest participant(s).

Controlling an oracle implies seeing both its inputs and outputs, therefore:

$$\frac{Adv\ \mathrm{ctl}^{off} O^f(\cdot)}{Adv \blacktriangleleft O^f(\blacktriangleright)} \qquad \frac{Adv\ \mathrm{ctl}^{nb} O^f(\cdot)}{Adv \blacktriangleleft^{nb} O^f(\blacktriangleright)} \qquad \frac{Adv\ \mathrm{ctl}^{bl} O^f(\cdot)}{Adv \blacktriangleleft^{bl} O^f(\blacktriangleright)} \tag{1}$$

Observation, off-line or non-blocking on-line control produce no different protocol behavior, thus guesses can go undetected. Yet, guessing using *blockable access* is also feasible, if a single oracle access suffices for verification.

The number of oracle accesses affects the feasibility of a guess. We specify lower bounds for *observes* and *controls*: $Adv \blacktriangleleft_b O^f(\cdot)$ and $Adv\ \mathrm{ctl}_b O^f(\cdot)$ mean that *Adv* observes at least b *distinct* independent outputs of $O^f(\cdot)$ (respectively, observes outputs of $O^f(\cdot)$ for b chosen inputs) over different protocol runs. Bounds on these relations are deduced from the protocol description. If $A \to B : H(N_A)$, then since N_A is randomly chosen, $Adv \blacktriangleleft_b O^g(\cdot)$ for any $b \leq |N_A|$ (set cardinality). However, if $A \to B : id_A, H(id_A.k_{AB})$, the oracle input is constant (and only partially visible, since k_{AB} is unknown), and we can only state $Adv \blacktriangleleft_1 O^g(\triangleright)$. Our goal is to conservatively warn for guessing attacks, thus we do not use upper bounds to rule out attacks, although the approach could be extended.

3 Rules for guessing

3.1 Outline of the approach

We formalize guessing attacks done under two distinct circumstances:

1) *Adv* knows the output of a function f computed on the secret (and optionally, known additional inputs). Using an oracle $O^f(\cdot)$ for f, *Adv* computes f on all possible secret values (with the known extra inputs), and verifies the guess comparing with the known output for the secret. Examples are: *Adv* knows $H(s)$, i.e., the output of the function $f(x) = H(x)$ on s, or *Adv* knows $m, MAC_s(m)$, i.e., the output of the $f(x, y) = MAC_x(y)$ on s and a second known input m. To verify the guess, f must be injective, otherwise more secrets can verify one output. In this case, we will formalize guessing using several outputs for the same secret (with different additional inputs).

2) *Adv* knows one or more outputs of an invertible function f computed on the secret and some additional, possibly unknown inputs. The adversary uses an oracle for the inverse of f and computes the inputs to the known output(s) for each possible value of the secret. A guess is verified using a known property that identifies the correct input(s) to f. This may be: (1) a relation to a known value (a known part of the input to f), (2) a relation between different parts of the same input, or, (3) if there are several inputs, a relation between them. For (1), knowing $E_s(id_A.m)$, *Adv* can guess s by checking for the known value id_A in the input (obtained by inverting the output, i.e. by decryption with all s). For (2), if *Adv* knows $E_s(m, m)$, he can check for which value of s the result of decryption (inversion) has identical halves (a relation between parts of the original input). For (3), knowing $E_s(H(m))$ and $E_s(m)$, *Adv* inverts both outputs and checks if the two inputs are related by means of H.

This way of verifying the guess is valid only if the inverse of f behaves as a pseudo-random function for a wrong value of s. Therefore, we allow this guessing rule only for encryption and decryption functions with keys dependent on s.

Our proposed approach works as follows: first, potential guessing opportunities are detected by matching them with one of the aforementioned situations, which are formalized in the guessing lemma of the next section. Next, oracle definition rules based on Dolev-Yao intruder capabilities are used to infer whether *Adv* has access to the required oracles. If so, we warn that guessing is feasible.

3.2 The Guessing Lemma

Definition 1. Given $\sigma \in \{0,1\}^k$, we call a function $f(\sigma, x)$ *distinguishing* in its first argument if there exists an algorithm $\mathcal{D}_f$, polynomial-time in k, that outputs a set $S = \{x_1, x_2, ..., x_{p(k)}\}$ such that the probability that there exists $s_1 \neq s_2$ such that $\forall x_i \in S \,.\, f(s_1, x_i) = f(s_2, x_i)$ is negligible, i.e., $Pr[f(s_1, x_i) = f(s_2, x_i), i = 1\,..\,p(k), s_1 \neq s_2] \leq v(k)$.

Here $p(k)$ is a polynomial in k and $v(k)$ is a negligible function, i.e., $\forall c \geq 0$ there exists k_c such that $\forall k \geq k_c \,.\, v(k) < k^{-c}$. For our calculus we use a notion that is more precise quantitatively, *strongly distinguishing* function *in q queries*:

Definition 2. Given $\sigma \in \{0,1\}^k$, we call a function $f(\sigma, x)$ *strongly distinguishing* in the first argument after q queries, if given any q distinct queries $\{x_1, x_2, ..., x_q\}$, $\forall s_1 \neq s_2$ the probability that $f(s_1, x_i) = f(s_2, x_i)$ for all $i = 1\,..\,q$ is at most 2^{-k}, i.e., $\forall s_1 \neq s_2 \,.\, Pr[f(s_1, x_i) = f(s_2, x_i), i = 1\,..\,q] \leq 2^{-k}$.

Note that any injective function with one argument is strongly distinguishing in one query, if we consider the second argument to be null.

The second definition allows us to give a quantitative bound on the number of attempts needed for guessing. Assuming we know q outputs of $f(s_1, x)$ for the unknown secret s_1 and $x \in \{x_1, x_2, ..., x_q\}$, then after q queries to $f(s, x)$ for each candidate value $s \in \{0,1\}^k$, in average only the correct secret s_1 will match all known outputs of f. With our definition of *distinguishing* and *strongly distinguishing* functions we do not aim to guarantee uniqueness of the secret, but to give guessing a probabilistic meaning which addresses the average case.

Example 1. Let $\sharp s\sharp$ denote a term obtained by concatenating something to s (to the left, right of s or both). Then $E_{\sharp s\sharp}(\cdot), D_{\sharp s\sharp}(\cdot)$ are *distinguishing* and *strongly distinguishing* in one query assuming that encryption and decryption with different keys performed on the same value cannot yield the same result. Also, $H(\sharp s\sharp)$ is *distinguishing* and *strongly distinguishing* in one query if H is collision-free on the argument range of $\sharp s\sharp$.

Example 2. Let $g(\sigma, x) = H(\sigma, x) \bmod 2^l$, $s \in \{0,1\}^k$. If H outputs more than l bits, one query is not sufficient to distinguish the secret. After q queries with $x \in \{x_1, x_2, ..., x_q\}$ we have $Pr[H(s_1, x) = H(s_2, x)] = 2^{-ql}$ for any $s_1 \neq s_2$. If the input space is $\{0,1\}^k$, the average number of values for which collisions occur after q queries is 2^{k-ql}, therefore g is *strongly distinguishing* in k/l queries.

To express our guessing rules, we first formalize the ability of the adversary to find relations between oracle observations or subterms thereof.

Definition 3. Given a function h and a list of terms α, we say there is a relation under h with arguments from α, denoted $R(h, \alpha)$, if the adversary can establish an equality $h(\beta) = \gamma$ such that: i) β, γ are terms constructed from the adversary knowledge and two disjoint subsets of terms from α, with at least one subset non-empty; ii) $h(\beta)$ is injective in at least one input that comes from α, with all other inputs kept constant.

This relation is used in the guessing lemma. Condition i) forces *Adv* to validate a guess by using at least one term deduced after the guess, while condition ii) avoids trivial identities with terms that can result from a wrong guess.

Lemma 1 (Guessing Lemma). Let $s \in \{0,1\}^k$ be a low-entropy value (i.e., 2^k computation steps are feasible), and f an *strongly distinguishing* function in q queries. The following guessing rules hold:

i) If $Adv \blacktriangleleft_{b_1} O^f(s, \blacktriangleright)$ and $Adv\ \mathrm{ctl}_{b_2} O^f(\cdot,\cdot)$, then *Adv* can guess s with q observations of $O^f(s,\cdot)$ and $q{\cdot}2^k$ queries to $O^f(\cdot,\cdot)$, i.e.

$$\frac{Adv \blacktriangleleft_{b_1} O^f(s, \blacktriangleright) \wedge Adv\ \mathrm{ctl}_{b_2} O^f(\cdot,\cdot)}{Adv \rightsquigarrow s} \quad \begin{matrix} b_1 \geq q \\ b_2 \geq q \cdot 2^k \end{matrix} \tag{2}$$

ii) If $Adv \blacktriangleleft_{b_1} O^{E_{f(s,\blacktriangleright)}}(\alpha)$, $Adv\ \mathrm{ctl}_{b_2}\{O^{D_{f(\cdot,\cdot)}}(\cdot), O^h(\cdot)\}$, and $R(h,\alpha)$, then *Adv* can guess s with q observations of $O^{E_{f(s,\cdot)}}(\cdot)$ and $q{\cdot}2^k$ queries to $O^{D_{f(\cdot,\cdot)}}(\cdot)$, $O^h(\cdot)$.

$$\frac{Adv \blacktriangleleft_{b_1} O^{E_{f(s,\blacktriangleright)}}(\alpha) \wedge Adv\ \mathrm{ctl}_{b_2}\{O^{D_{f(\cdot,\cdot)}}(\cdot), O^h(\cdot)\} \wedge R(h,\alpha)}{Adv \rightsquigarrow s} \quad \begin{matrix} b_1 \geq q \\ b_2 \geq q{\cdot}2^k \end{matrix} \tag{3}$$

iii) If $Adv \blacktriangleleft_{b_1} O^{E_{f(s,\blacktriangleright)}}(\alpha_i)$, with distinct α_i, $Adv\ \mathrm{ctl}_{b_2}\{O^{D_{f(\cdot,\cdot)}}(\cdot), O^h(\cdot)\}$, and $R(h,\overline{\alpha})$, with $\overline{\alpha} = (\alpha_1, \ldots, \alpha_n)$, then *Adv* can guess s with q observations of $O^{E_{f(s,\cdot)}}(\cdot)$ and $q{\cdot}2^k$ queries to $O^{D_{f(\cdot,\cdot)}}(\cdot)$, $O^h(\cdot)$.

$$\frac{Adv \blacktriangleleft_{b_1} O^{E_{f(s,\blacktriangleright)}}(\alpha_i) \wedge Adv\ \mathrm{ctl}_{b_2}\{O^{D_{f(\cdot,\cdot)}}(\cdot), O^h(\cdot)\} \wedge R(h,\overline{\alpha})}{Adv \rightsquigarrow s} \quad \begin{matrix} b_1 \geq q \\ b_2 \geq q{\cdot}2^k \end{matrix} \tag{4}$$

Proof sketch. In case i) by Def. 2, for q observations of $O^f(s,\cdot)$, in average only one s verifies the input-output relation, so $b_1 \geq q$ suffices. Thus, *Adv* can find s by making queries to $O^f(\cdot,\cdot)$ with all 2^k values of s and the q observed inputs, then verify them against the q observed outputs for $O^f(s,\cdot)$.

A sufficient bound on queries is $q{\cdot}2^k$, however fewer queries are needed since each query reduces the number of candidates for s in average by a factor of $2^{k/q}$.

Cases ii) and iii) are similar, but require the additional queries to $O^h(\cdot)$. If $R(h,\alpha)$ holds for the encryption input, this confirms the secret.

Case (i) is a direct match of the oracle output. Case (ii) matches the input and part of the decryption output, e.g., when Adv knows $\{m, E_s(\sharp H(m)\sharp)\}$ or a relation between parts of the decrypted output, e.g., if Adv knows $E_s(\sharp m\sharp H(m)\sharp)$. Adv controls the decryption oracle and thus can check for $H(m)$ in the decryption result for all values of the secret. Case (iii) matches different inputs to the same oracle, e.g., when Adv knows $\{E_s(\sharp m\sharp), E_s(\sharp H(m)\sharp\}$.

Corollary 1. In case i) of the Guessing Lemma, if $O^f(s,\cdot)$ has no random inputs, or Adv controls the oracle $O^f(s,\cdot)$ (i.e., can choose all inputs, the q observations become queries), then Adv can do pre-computed dictionary guessing:

$$\frac{Adv \blacktriangleleft_{b_1} O^f(s,\blacktriangleright) \wedge Adv\ \mathrm{ctl}_{b_2} O^f(\cdot,\cdot)}{Adv \rightsquigarrow_D s} \quad \frac{Adv\ \mathrm{ctl}_{b_1} O^f(s,\cdot) \wedge Adv\ \mathrm{ctl}_{b_2} O^f(\cdot,\cdot)}{Adv \rightsquigarrow_D s} \quad \begin{array}{l} b_1 \geq q \\ b_2 \geq q{\cdot}2^k \end{array} \tag{5}$$

Example 3. Let E be deterministic encryption and H a hash function. Then, $E_k(\cdot), H(\cdot)$, and $E.(\alpha)$ are injective, and thus *strongly distinguishing* in one query. By Corollary 1: $\dfrac{Adv \blacktriangleleft_1 O^q(s) \wedge Adv\ \mathrm{ctl}^{nb}\ O^q(\cdot)}{Adv \rightsquigarrow_D s}$

$$\frac{Adv \blacktriangleleft_1 O^{E_k}(s) \wedge Adv\ \mathrm{ctl}^{nb}\ O^{E_k}(\cdot)}{Adv \rightsquigarrow_D s} \quad \frac{Adv \blacktriangleleft_1 O^{E(\alpha)}(s) \wedge Adv\ \mathrm{ctl}^{nb}\ O^{E(\alpha)}(\cdot)}{Adv \rightsquigarrow_D s} \tag{6}$$

We have used ctl^{nb} which is weaker than ctl^{off} but sufficient to verify a guess.

4 Case studies

4.1 Anderson-Lomas Protocol

We focus on protocols which expose verifiers that fit more than one secret, a case not previously addressed using guessing rules. The Anderson-Lomas protocol [2] is relevant for the ingenuity in constructing password verifiers by using a *collisionful* hash function, i.e., a function $q(k,x)$ for which given x one can find $k' \neq k$ such that $q(k,x) = q(k',x)$. The protocol description is as follows:

(1) $A \rightarrow B : g^{r_A}$ (3) $A \rightarrow B : H(MAC_{pw} g^{r_A r_B} \bmod 2^m, g^{r_A r_B})$

(2) $B \rightarrow A : g^{r_B}$ (4) $B \rightarrow A : H(MAC_{H(pw)} g^{r_A r_B} \bmod 2^m, g^{r_A r_B})$

Here, $g^{r_A r_B}$ is the regular key from the Diffie-Hellman-Merkle key exchange protocol and pw is the password shared between A and B while m is a fixed constant suggested to be $n/2$ if the size of the password space is 2^n.

Let Adv play the role of B and consider the oracle $O^f(\cdot,\cdot)$, with $f(x,y) = H(MAC_x(y) \bmod 2^m, y)$. If the secret has k bits then f is *strongly distinguishing* in k/m queries. By choosing g^{r_A}, Adv can compute $g^{r_A r_B}$ and therefore knows both the input and output of O^f. Then, we have

$$\frac{Adv \blacktriangleleft_{b_1} O^f(s,\blacktriangleright) \wedge Adv\ \mathrm{ctl}_{b_2} O^f(\cdot,\cdot)}{Adv \rightsquigarrow pw} \quad \begin{array}{l} b_1 \geq \frac{k}{m} \\ b_2 \geq \frac{k}{m} \cdot 2^k \end{array} \tag{7}$$

according to case (i) of the guessing lemma which allows us to formalize the attack originally presented in [2] and explain it using a general guessing rule.

4.2 The Norwegian ATM

With our calculus we formalize attacks in a Norwegian ATM system, shown to be flawed in [10]. The system attempts to increase password security by hiding the verifier. Cards store the PIN encrypted with a bank key BK, truncated to 16 bits: $\lfloor DES_{BK}(PIN)\rfloor_{16}$ (simplified, since the PIN is not encrypted directly).

To find the PIN of a stolen card, Adv cannot guess the PIN off-line without BK, since for each PIN about 2^{40} of 2^{56} DES keys match. However, [10] presents a more subtle attack. Adv gets *several honest* cards from the same bank. Each known PIN reduces the number of candidate keys by a factor of 2^{16}. On average, 4 honest cards suffice to find BK, and then guess the PIN of the stolen card.

The services provided by the bank and ATM to a user are summarized below:

$$\begin{array}{ll} \textit{Card issuing stage:} & \textit{Bank} \rightarrow \textit{User} : \lfloor DES_{BK}(PIN)\rfloor_{16}, PIN \\ \textit{PIN change procedure:} & \textit{User} \rightarrow \textit{ATM} : \lfloor DES_{BK}(PIN_{old})\rfloor_{16}, PIN_{old}, PIN_{new} \\ & \textit{ATM} \rightarrow \textit{User} : \lfloor DES_{BK}(PIN_{new})\rfloor_{16} \end{array}$$

We assume PIN and card (holding $\lfloor DES_{BK}(PIN)\rfloor_{16}$) are issued securely, otherwise a Dolev-Yao adversary could get the PIN directly from the protocol.

Since $\log_2 |PIN| < 16$, $\lfloor DES_{BK}(\cdot)\rfloor_{16}$ is *strongly distinguishing* in one query and Adv can guess the PIN using the PIN change procedure as oracle:

$$\frac{\dfrac{Adv \text{ knows } \lfloor DES_{BK}(PIN)\rfloor_{16}}{Adv \blacktriangleleft_1 O^{\lfloor DES_{BK}(\cdot)\rfloor_{16}}(PIN)} \quad \dfrac{Adv \rightarrow PIN_{new} \quad ATM \rightarrow \lfloor DES_{BK}(PIN_{new})\rfloor_{16}}{Adv \text{ ctl}^{nb} O^{\lfloor DES_{BK}(\cdot)\rfloor_{16}}(\cdot)}}{Adv \rightsquigarrow_D PIN} \tag{8}$$

In reality, the PIN is encrypted concatenated with a card-specific value CV. Thus, changing the PIN no longer controls the encryption oracle. To find the PIN, Adv must simulate the oracle himself, and for this, BK must be known:

$$\frac{\dfrac{Adv \text{ knows } \lfloor DES_{BK}(PIN.CV)\rfloor_{16}}{Adv \blacktriangleleft_1 O^{\lfloor DES_{BK}(\cdot)\rfloor_{16}}(PIN.CV)} \quad \dfrac{Adv \text{ knows } BK}{Adv \text{ ctl}^{off} O^{\lfloor DES_{BK}(\cdot)\rfloor_{16}}}}{Adv \rightsquigarrow PIN} \tag{9}$$

For this goal, Adv needs to *observe* an oracle output on BK. One possibility is in the card issuing stage. Let $f(\sigma, x) = \lfloor DES_{\sigma}(x)\rfloor_{16}$. Then, f is *strongly distinguishing* in 4 queries since the DES key has 56 bits, and we have:

$$\frac{\dfrac{Adv \text{ knows } PIN.CV_{1..4}, \lfloor DES_{BK}(PIN.CV_{1..4})\rfloor_{16}}{Adv \blacktriangleleft_4 O^{f}(BK, \cdot)} \quad Adv \text{ ctl}^{off} O^{f}(\cdot, \cdot)}{Adv \rightsquigarrow BK} \tag{10}$$

Another option comes again from controlling the PIN change procedure. Let $g(\sigma, x) = \lfloor DES_{\sigma}(CV.x)\rfloor_{16}$ for a card of the adversary with value CV. Then,

$$\frac{Adv \text{ ctl}^{nb} O^{g}(BK, \cdot) \quad Adv \text{ ctl}^{off} O^{g}(\cdot, \cdot)}{Adv \rightsquigarrow BK} \tag{11}$$

This attack fits the real-world situation where the adversary can change his own PIN and is new to the best of our knowledge. In [10], only the attack using several cards from the bank (to guess all DES key bits) is given. Moreover, our calculus distinguishes two ways to find BK. The attacks illustrate both dictionary and and pre-computed dictionary guessing.

5 Conclusions

We have introduced a calculus based on oracles for detecting guessing attacks, with rules that supplement the deductions of a Dolev-Yao intruder. The rules are based on *observes* and *controls* relations between the adversary and oracles. Conceptually separating generating the verifier from verifying the guess justifies consideration of on-line guessing attacks which may not be detected and blocked. The calculus can be used in a mixed on-line/off-line setting. Our guessing rules also handle protocols with verifiers that match more than one secret. In this case, guessing has a probabilistic meaning, and our rules give sufficient bounds on the number of observations for successful attacks. We formalize the known flaws in the Anderson-Lomas and Norwegian ATM protocols in this framework. For the ATM protocol, our calculus finds new attacks based on a PIN change procedure and on the use of the ATM as encryption oracle.

Acknowledgments Thanks to Cas Cremers who helped clarify a first writeup of our approach and to the anonymous reviewers for their valuable comments.

References

1. M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Proc.FoSSaCS 2006*, *LNCS* 3921, pp. 398–412.
2. R. J. Anderson and T. M. A. Lomas. Fortifying key negotiation schemes with poorly chosen passwords. *Electronics Letters*, 30(13):1040–1041, July 1994.
3. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc.* 12^{th} *ACMConf. on Computer and Communications Security*, pp. 16–25, 2005.
4. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In 14^{th} *IEEE Computer Security Foundations Workshop*, pp. 82–96, 2001.
5. R. Corin, J. M. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. WISP 2004*, pp. 47–63.
6. R. Corin, S. Malladi, J. Alves-Foss, and S. Etalle. Guess what? here is a new tool that finds some new guessing attacks. In *Proc. WITS 2003*, pp. 62–71.
7. S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *Proc.* 17^{th} *IEEE Computer Security Foundations Workshop*, pp. 2–15, 2004.
8. Y. Ding and P. Horster. Undetectable on-line password guessing attacks. *Operating Systems Review*, 29(4):77–86, 1995.
9. P. H. Drielsma, S. Mödersheim, and L. Viganò. A formalization of off-line guessing for security protocol analysis. In *Proc. LPAR 2004*, *LNCS* 3452, pp. 363–379.
10. K. J. Hole, V. Moen, A. N. Klingsheim, and K. M. Tande. Lessons from the Norwegian ATM system. *IEEE Security and Privacy*, 5(6):25–31, 2007.
11. G. Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1):83–98, 2004.