Secure upgrade of hardware security modules in bank networks*

Riccardo Focardi[1] and Flaminia L. Luccio[1]

Università Ca' Foscari Venezia,
{focardi,luccio}@dsi.unive.it

Abstract. We study the secure upgrade of critical components in wide networked systems, focussing on the case study of PIN processing Hardware Security Modules (HSMs). These tamper-resistant devices, used by banks to securely transmit and verify the PIN typed at the ATMs, have been shown to suffer from API level attacks that allow an insider to recover user PINs and, consequently, clone cards. Proposed fixes require to reduce and modify the HSM functionality by, e.g., sticking on a single format of the transmitted PIN or adding MACs for the integrity of user data. Upgrading HSMs worldwide is, of course, unaffordable. We thus propose strategies to incrementally upgrade the network so to obtain upgraded, secure subnets, while preserving the compatibility towards the legacy system. Our strategies aim at finding tradeoffs between the cost for special "guardian" HSMs used on the borderline between secure and insecure nodes, and the size of the team working in the upgrade process, representing the maximum number of nodes that can be simultaneously upgraded.

Keywords: Security APIs, PIN processing, Hardware Security Modules, Upgrade strategies.

1 Introduction

Automated Teller Machines (ATMs) verify the user's Personal Identification Number (PIN) by sending it to the issuing bank on an international bank network. During their journey, PINs are decrypted and re-encrypted on the traversed network switches by special tamper-resistant devices called Hardware Security Modules (HSMs). As shown in figure 1, the keypad itself is an HSM that performs the first PIN encryption with a symmetric key k_1 shared with the neighbour acquiring bank. Before the encrypted PIN is routed to the next network node, it is passed to the HSM in the switch that decrypts and re-encrypts it with another key k_2, shared with the destination node, and so on.

In the last years, several API-level attacks have been discovered on these HSMs [5, 6, 9]. The attacker is assumed to be an insider gaining access to the

* Work partially supported by Miur'07 Project SOFT: "*Security Oriented Formal Techniques*".

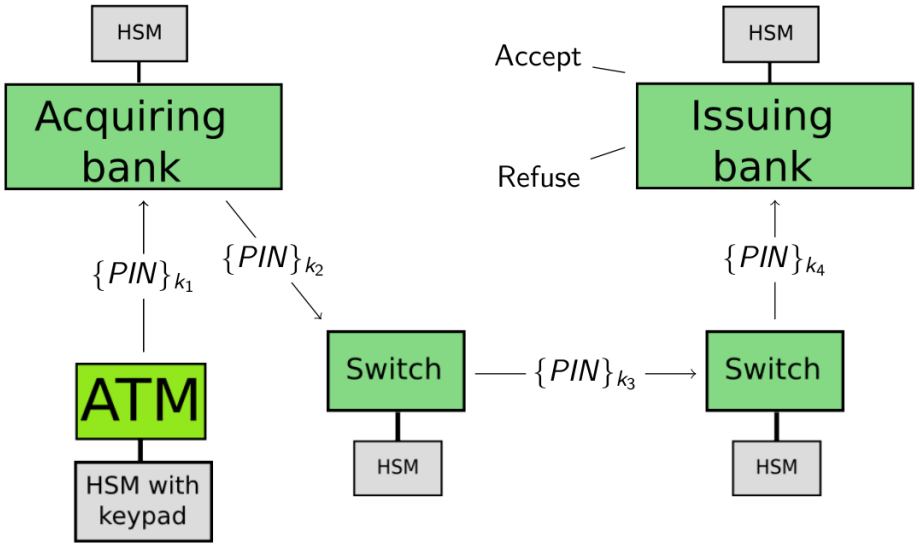

Fig. 1. PIN processing infrastructure.

HSM at some bank switch. Then, by performing subtle sequences of API calls he is able to deduce the value of the PIN. As an example, the so-called *dectab attack* is based on manipulating public user data given as input to the PIN verification API at the issuing bank HSM. One of these data is a decimalization table that maps an intermediate hexadecimal representation of the user PIN into a decimal number. By modifying the way numbers are decimalized and by observing if this affects the result of the verification, the attacker can deduce which are the actual PIN digits: if the original dectab maps, e.g., A into 0 and the attacker modifies it so to map A into 1 causing a failure in the PIN verification API, then he is guaranteed that a 0 digit was occurring in the decimalized user PIN. If it were not the case, PIN verification result would be unaffected by the change in the *dectab*. The attack goes on using another public parameter, the *offset*, to reconstruct the whole PIN code. The interested reader is referred to, e.g., [8, 9, 12, 20], for more detail.

The interest in these API-level attacks has recently increased [1, 2] and it becomes more an more plausible that some of them have been really exploited by malicious insiders to steal thousands of user PINs. This has motivated research on formal methods for analysing PIN recovery attacks and API-level attacks in general [20]. In particular, in [8] we have proposed a language-based setting for analysing PIN processing API via a type-system. We have formally modelled existing attacks, proposed some fixes and proved them correct via type-checking. These fixes typically require to reduce and modify the HSM functionality by, e.g., sticking on a single format of the transmitted PIN or adding MACs for the

integrity of user data. Notice, in fact, that the above mentioned attack is based on the absence of integrity on public user data such as the *dectab* and the *offset*.

There are crucial difficulties when trying to implement these fixes on a real bank network: first of all, upgrading the bank network HSMs worldwide is complex and very expensive; moreover, adding improved APIs in the HSMs is not enough: we also need to remove old flawed APIs if we want to stop the attacks. This typically makes upgraded HSMs incompatible with the old ones, for example when one switch expects a MAC and the previous non-upgraded one cannot provide it. These difficulties can be circumvented at the price of loosing some security: in [11] we have proposed a low-impact, easily implementable fix requiring no hardware upgrade which makes attacks 50000 times slower, but yet not impossible.

Our contribution. We propose a novel way of upgrading critical components in wide networked system that mitigates the above discussed difficulties. In particular, we propose strategies to incrementally upgrade the network so to obtain upgraded, secure subnets while preserving the compatibility towards the legacy system. This should make the upgrading process more appealing to banks, as they might decide to invest money for upgrading part of the system still maintaining the interoperability with the non-upgraded part. Of course, PINs travelling through non-upgraded subnets would be exposed to the same attacks as before, but PINs traversing secured subnets would be robust against API-level attacks.

To guarantee the compatibility with the old system we propose the adoption of special *borderline* HSMs translating from/to the old protocol. These HSMs are temporarily placed on the borderline between upgraded and non-upgraded nodes and can be thought as the 'union' of the old and the new HSMs, thus supporting the functionalities of both and being able to translate from/to the old protocol. Of course, this hardware is far from being secure but it can be used to temporarily keep the network working while the upgrade is performed. Our strategies aim at finding tradeoffs between the cost for borderline HSMs and the size of the team working in the upgrade process, representing the maximum number of nodes that can be simultaneously upgraded: since HSMs are quite expensive, it might make sense to have bigger teams of technicians able to upgrade a whole subtree in one shot, with no need of placing many borderline HSMs around.

We formally state the HSM upgrading problem and we prove that, when only one technician is present, it is strictly related to the Connected Monotone Decontamination (CMD) problem [3], where a team of agents aims at decontaminating a graph and capturing an intruder. The analogy is that decontaminated nodes should never be directly in contact with contaminated ones to avoid the intruder re-contaminating them, thus some agents must stay on the borderlines to 'guard' the decontaminated subnetwork. This is similar to what happens when placing a borderline HSM between upgraded and non-upgraded nodes. We then generalize the optimum algorithm for the CMD problem to our setting, also considering many technicians. We prove the new algorithm correct and we show how it can be applied to estimate upgrading cost for a real bank (sub)network.

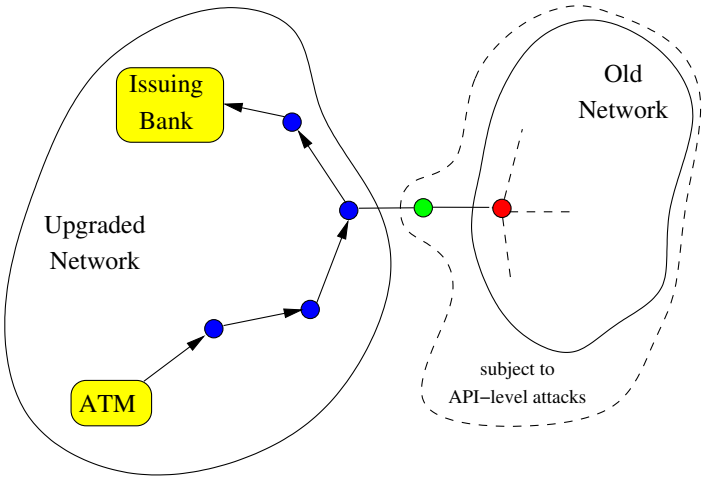

Fig. 2. Borderline HSM on a link.

Paper structure. The paper is organized as follows: in section 2 we formalize the HSM upgrading problem and we prove it equivalent to the CMD problem, up to one borderline HSM; in section 3, we give an algorithm for upgrading HSMs which is parametrized by the size of the technician team; in section 4 we give an example of application of the algorithm to estimate the upgrading cost on a network; finally, in section 5 we give some concluding remarks and we discuss future work.

2 The HSM upgrading problem

Given a bank network, we want to study strategies for incrementally upgrading HSMs while keeping the network functionality up, apart from the lapse of time in which HSMs are physically substituted. While applying an upgrading technique, only part of the HSMs will be upgraded. As we have mentioned in the introduction, in this upgraded hardware we certainly want to remove old flawed functionalities, thus it will not be able, in general, to support the old protocol and communicate with non-upgraded HSMs. To this aim, the technicians performing the upgrade can place special *borderline* HSMs translating from/to the old protocol.

We briefly discuss two possible settings, depending on whether these special HSMs are placed on *links* as in figure 2, or on *switches*, as in figure 3. The former setting makes all the paths in the upgraded network secure, but in case we have many links towards the non-upgraded network it would require to install many borderline HSMs thus becoming more complicated and expensive. The latter setting is cheaper and more flexible: once an HSM becomes borderline all the neighbouring switches can be indifferently upgraded or not. Notice, however, that the paths passing through borderline HSMs in the upgraded network are

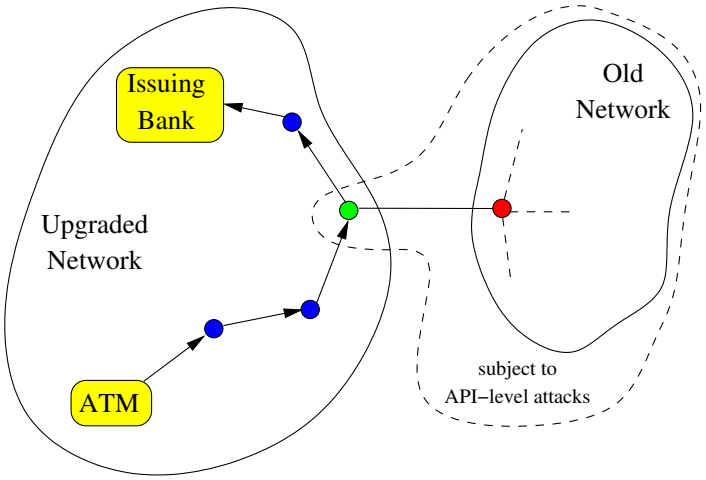

Fig. 3. Borderline HSM on a node.

still subject to the attacks. This is because borderline nodes still support old, flawed functionalities. This setting is cheaper than the former, but the same degree of security is only achieved later on, when more nodes will be upgraded. Given that we believe the cost is likely to be the major issue we prefer to analyse this latter setting and leave the former as a future work.

Another issue to be considered is which network *topology* we can expect to be faced with, when starting an upgrade. In the USA, real systems are composed of regional networks shared by different banks and are hierarchically connected through switches of one of the existing national networks [13]. In our opinion, a reasonable assumption to make is that the updates start inside regional networks which will be composed of different trees, each representing a different bank network, rooted at the issuing bank. It seems plausible that different banks will upgrade their hardware independently, thus we imagine they will first install a sufficient number of borderline switches to 'separate' their network from the other bank networks in the same region. We assume this preliminary separation as done and we focus on single trees representing single bank networks, rooted at the issuing bank node and having all the ATMs as leaves.

We can now formally state the HSM upgrading problem as an algorithmic problem on trees.

The HSM upgrading problem. We consider an initially *non-upgraded* tree network that has to be *upgraded* by a set of *technicians* arbitrarily moving inside the network and possibly placing *borderline HSMs*. All technicians and borderline HSMs are placed in one initial node. A technician can move only in a *connected* way, i.e., from a node to another along an edge, in doing so she upgrades the traversed nodes. Borderline HSMs can only be moved by one technician; in that case the HSM and the technician move together along edges. The technician

moves such special hardware when all the neighbouring nodes, except the destination one, have been upgraded. In this case, in fact, the borderline HSM is not any more needed. We implicitly assume that this hardware, in the presence of one technician and with all neighbouring nodes upgraded is switched off, making the node upgraded. At the end of the computation all the nodes are upgraded.

An *HSM upgrading strategy* is a sequence of moves that will upgrade an initially non-upgraded network. While devising efficient updating techniques we consider different parameters that we would like to optimize:

1. the number B of available borderline HSMs: each of them has a cost which may be very high, typically around 10000$, thus B has to be minimized;
2. the number U of technicians in the upgrading team: each technician can upgrade one HSM in one node, so having many technicians allows for simultaneous upgrading of subsets of nodes; the salary we pay to the technicians is proportional to this parameter, thus even U should me minimized;

Definition 1 (HSM upgrading number). *It is the minimal number of borderline HSMs needed to solve the HSM upgrading problem on a given tree T and with a given number U of technicians. We note it $uhn(T, U)$.*

2.1 Upgrading vs. decontaminating

We now show that the HSM upgrading problem with a single technician is very strictly related to the *Connected Monotone Decontamination* (CMD) problem, formalized below.

The CMD problem [3]. We consider an initially *contaminated* network that has to be *decontaminated* by a set of agents (or *searchers*) arbitrarily moving inside the network. All nodes and edges are initially *contaminated*, except for the initial node where all the agents are located, which is *guarded*. An agent can move only in a *connected* way, i.e., from a node to another along an edge. The network contains an intruder that contaminates the nodes and edges it traverses. The intruder cannot traverse guarded nodes, otherwise it would be captured by the agents, but we assume that he will immediately recontaminate nodes or edges that are left unguarded. In the *node decontamination problem* a node is guarded when it contains at least one agent, clean when an agent has been on the node and all the neighbouring nodes are clean or guarded, and contaminated otherwise. At the end of the computation all the nodes of the network are simultaneously clean. In the *edge decontamination problem* an edge becomes *clean* whenever an agent passes through it, and its starting node is either clean or guarded. At the end of the computation all the nodes and edges of the network are simultaneously clean.

A *decontamination (or search) strategy* is a sequence of moves that will clear an initially contaminated network. A strategy is *monotone*, if a decontaminated node or edge is never recontaminated. Finally, the strategy has to be efficient, where efficiency is measured in terms of the size of the agent team. A strategy is optimal if the size of the team is minimal.

Definition 2 (Connected search number). *It is the minimal team size needed to solve the edge CMD problem on a given network G, and is noted $csn(G)$.*

The problem of finding an optimal strategy has been studied in some specific topologies such as trees [3]. In the same work it is also proved that, for trees, a non-monotone solution, i.e., allowing re-contamination of nodes, would not reduce the optimal number of required agents.

Originally, the decontamination problem has been introduced in [7, 17] and has been extensively studied in the literature under the term *graph search* (e.g., see [10, 14–16, 18]). The difference with the *CMD* problem is that searchers do not necessarily move in a connected way, i.e., may 'jump' from one node to another. In the HSM upgrading problem technicians do not travel via the network they repair, but via lorries or cars. In an ideal setting we would thus have to think our scenario as a composition of two networks: the bank network overlapped with a geographical network where technicians physically move. Another approach could consist of considering only the bank network where the travelling cost is added by assigning weights to the edges (where weights represent physical distances), and by letting searchers jump from one node to the other. It is interesting to note that for any non-weighted tree T with n nodes the ratio between the connected search number $csn(T)$ and the regular search number $sn(T)$ (i.e., for agents that may jump) is bounded by, $csn(T)/sn(T) \leq 2$ (see [4]), i.e., any optimal connected strategy will require at most twice the number of agents than a non-connected one (at least in the non-weighted case). We thus leave the two above mentioned extensions as a future work and for simplicity, in this work we concentrate on the scenario in which we do not consider the technician motion as a parameter to optimize.

Equivalence of HSM upgrading and CMD. We now prove that the HSM upgrading problem solved using the minimal number of borderline HSMs and a single technician is equivalent, up to one extra agent, to the problem of finding a connected monotone technique for the edge decontamination of the tree network using the smallest number of agents.

Theorem 1. *Given a tree T, we have $uhn(T, 1) \leq csn(T) \leq uhn(T, 1) + 1$.*

Proof. Let us first assume we have solved the HSM upgrading problem in a tree using the minimal number $uhn(T, 1)$ of borderline HSM and a single technician. We show that the adopted strategy can be mapped into a valid strategy for the CMD problem with a number of agents equal to $uhn(T, 1) + 1$. We can see upgraded and non-upgraded HSMs respectively as decontaminated and contaminated nodes. Moreover, we see each borderline HSM and the (single) technician as agents. We now show that moves of the technician and of borderline HSMs (that are anyway transported by the technician) are correctly mapped into agent moves.

We first consider the technician moving alone, bringing no HSM. The only crucial case is when the node where she moves from has no borderline HSM on it meaning, in the other problem, that she is the only agent on the node. In this

case we are guaranteed that all the neighbouring nodes are either upgraded or borderline HSMs, as an upgraded node (the one where the technician is moving from) can never be directly connected to a non-upgraded one. The corresponding move in the CMD problem can be safely performed.

We now consider the technician moving with some borderline HSMs. If at least one HSM is left on the node the move can be safely simulated in the CMD problem, as at least one agent will stay on the originating node, guarding it. If all the borderline HSMs are moved by the technician we are in a situation similar to the one where the technician alone is moving: all the neighbouring nodes need to be either upgraded or borderline HSMs, except the destination node which, if needed, will be protected by one of the incoming borderline HSMs. This is perfectly safe in the CMD problem, as the moving agents will guard the destination node. The important thing is that the node left alone is protected by the upgraded/guarded neighbouring nodes.

When all the nodes are upgraded in the HSM upgrading problem they will correspondingly be decontaminated in the CMD problem, thus the strategy in the former problem is also a strategy in the latter, with a number of agents equal to $uhn(T, 1) + 1$. This implies $csn(T) \leq uhn(T, 1) + 1$.

Similarly, let us now assume we have solved the CMD problem with $csn(T)$ agents. We show that the adopted strategy can be mapped into a valid strategy for the HSM upgrading problem with one technician and with a number of borderline HSMs equal to $csn(T)$. We can see decontaminated and contaminated nodes as upgraded and non-upgraded HSMs, respectively. Moreover, we see each agent as a borderline HSM. We now show that moves of the agents are correctly mapped into borderline HSM moves, via the technician. Notice that no agent is mapped into the technician, so the position of the technician on the tree is immaterial. Notice also that, at any point of the computation, all upgraded/borderline HSMs are on a connected subtree of the network, given that agents start from a single initial node, they move in a connected way and HSMs can never be downgraded. Thus, we can freely move the technician on this upgraded/borderline nodes reaching all the borderline HSMs.

Now, any agent move is simulated by the technician reaching the corresponding borderline HSM and moving it in the destination node. The only crucial case is when the node which is left has no other agents guarding it. In this case, however, we are guaranteed that all of the neighbouring nodes, except the destination one, are upgraded or guarded, since an agent cannot allow a decontaminated node to be recontaminated. Thus, the corresponding borderline HSM can be safely removed from the node.

When all the nodes are decontaminated in the CMD problem they will correspondingly be upgraded or borderline in the HSM upgrading problem. At this point the technician can go around an collect all the borderline HSMs 'deactivating' them. Thus the strategy in the former problem is also a strategy in the latter, with a number of borderline HSMs equal to $csn(T)$. This implies $uhn(T, 1) \leq csn(T)$. ∎

3 Upgrading strategies

In this section we show different techniques to solve the HSM upgrading problem. We first provide strategies which aim at minimizing B, i.e., the number of borderline switches used, and then present trade-offs between B and U, i.e., the number of nodes U that can be simultaneously upgraded by a team of technicians.

Case $\mathbf{U = 1}$. Nodes are sequentially updated by a unique technician. As we have proved in the previous section, in this case this problem is equivalent to the CMD problem where agents are seen as borderline HSMs. We thus slightly modify the algorithm presented in [3] in order to solve our problem.

The main idea is to first compute, from every possible starting point, i.e., every node of the tree T, the minimal number of borderline switches required for the solution of the upgrading problem. Then, one of the nodes with minimal value is chosen as starting point and the actual upgrading algorithm is applied. In order to compute this minimal value two different rules (similar to the ones of [3]) need to be applied. For each edge $\{x, y\}$, we first compute $\lambda_x(x, y)$, i.e., the minimal number of borderline HSMs necessary for the upgrade of the tree rooted at y while arriving from x.

Rule 1 for computing minimal number of borderline HSMs on an edge

1. An edge $e = \{x, y\}$ leading from a node x to a leaf y requires only the technician moving from x to y to upgrade the HSM on y, thus $\lambda_x(x, y) = 0$;
2. An edge $e = \{x, y\}$ leading from a node x to a node y that has other k out-going edges requiring $l_1, \ldots, l_k$ borderline HSMs, with $l_1 \geq l_2 \geq, \ldots, \geq l_k$, requires $\lambda_x(x, y) = \max\{l_1, l_2 + 1, 1\}$ borderline HSMs.

From this value we can then compute the minimal number of borderline HSMs which are necessary for the upgrade of the tree starting from any node. The two rules are:

Rule 2 for computing minimal number of borderline HSMs on a node

1. A leaf y requires a number of borderline HSMs defined by $\lambda_y(y, x)$ to move to node x; in fact, being a leaf we do not need to leave any borderline HSM on y;
2. A node x that has h out-going edges each respectively requiring $l_1, \ldots, l_h$ borderline HSMs, with $l_1 \geq l_2 \geq, \ldots, \geq l_h$, needs $\max\{l_1, l_2 + 1\}$ borderline HSMs.

We now apply Rules 1 and 2 and then choose one of the nodes, say z, requiring a minimal number B of borderline HSMs. Similarly to the algorithm of [3],

consider T_z, the tree rooted at z and, for each node y of T_z, order its children by increasing values assigned by λ_z to the related links. Apply then the following:

Algorithm UPGRADE
1. Start at z with B borderline HSMs and a technician;
2. Traverse T_z in pre-order (by increasing values of λ_z). While moving from a node x to a node y use $\lambda_x(x, y)$ borderline HSMs and a technician, while returning to x bring back $\lambda_x(x, y)$ borderline HSMs and a technician.

Example 1. Let us now show how to apply Rules 1, 2 and Algorithm UPGRADE on a small example. Assume we have a tree T of 6 nodes called a, b, c, d, e, f (see figure 4). We first compute the values associated to the edges, i.e., we apply Rule 1. E.g., consider node d. As d is a leaf by Rule 1.1 we have $\lambda_b(b, d) = 0$, that is there is just a technician moving from b to d. The same holds for the other leaves e and f, thus $\lambda_c(c, e) = 0$, and $\lambda_c(c, f) = 0$. Consider now node a and edge $\{a, b\}$ and apply Rule 1.2. Node b has only one other out-going edge, i.e., $\{b, d\}$ that requires $l_1 = \lambda_b(b, d) = 0$ borderline HSMs. Thus, edge $\{a, b\}$ requires $\lambda_a(a, b) = \max\{l_1 = 0, 1\} = 1$ borderline HSM. That is, path $\{a, b, d\}$ is a chain thus we need at least one borderline HSM to upgrade it. On the other hand, if we consider edge $\{a, c\}$ and apply Rule 1.2 we have that node c has two other out-going edges, i.e., $\{c, e\}$ and $\{c, f\}$ that respectively require $l_1 = \lambda_c(c, e) = 0$ and $l_2 = \lambda_c(c, f) = 0$ borderline HSMs. Thus, edge $\{a, c\}$ requires $\lambda_a(a, c) = \max\{l_1 = 0, l_2 = 0 + 1 = 1, 1\} = 1$ borderline HSM. To compute the minimal number of borderline HSMs on a node we apply Rule 2. E.g., consider a leaf d. It requires a number of borderline HSMs defined by $\lambda_d(d, b) = 1$ to move to node b, thus we write 1 inside node d. Consider now node c that has 3 out-going edges each respectively requiring $l_1 = 1, l_2 = 0, l_3 = 0$ borderline HSMs, thus c needs $\max\{l_1 = 1, l_2 + 1 = 0 + 1 = 1\} = 1$ borderline HSM.

After having applied Rules 1 and 2 to all the nodes, we choose one of them requiring a minimal number of borderline HSMs, e.g., c requiring 1 HSM. We consider T_c, the tree rooted at c and order its children by increasing values assigned by λ_c to the related links, e.g., we consider node e, then f, then a, and finally apply Algorithm UPGRADE starting from c with 1 borderline HSMs and a technician.

Figure 5 shows an application of Algorithm UPGRADE starting from node c. Black nodes represent updated HSMs. Nodes surrounded by squares represent borderline HSMs, the lady represents the technician. At the end the technician goes back to c.

We now prove the correctness and the optimality of algorithm UPGRADE.

Theorem 2. *Algorithm* UPGRADE *together with Rule 1 and Rule 2 correctly and optimally solve the HSM upgrading problem in a bank tree network T.*

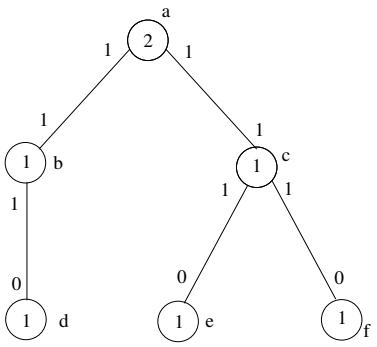

Fig. 4. Application of Rule 1 and 2.

Proof. The general correctness directly derives from the one of the rules and the algorithm presented in [3]. In particular, we first have to prove that an optimal monotone strategy for HSM upgrading with a single starting point on any tree T exists. Then, we choose the entry point that minimizes the number of required HSMs. Another key point for the correctness of the strategy is the property that the minimal number of required HSMs from a node x is given by the minimal number between the biggest minimal value of a son of x and the second biggest minimal value plus one. These values will correspond to the edge labels and will imply a specific ordering in the strategy visit. Finally, we prove that using the minimal number of HSMs, we can upgrade the system as moving back and forth with such number of HSMs does not leave parts of the network unprotected.

The existence of an optimal monotone strategy for CMD on any tree T has been proved in [3] using a variation of the proof of [15] for the non-monotone setting with the addition of the single starting point (i.e., the initially guarded node). The main idea that we borrow is that it is possible to build a progressive connected crusade (i.e., a sequence of moves where a new single un-traversed edge is added at each step, leading to the visit of the whole network and using only connected sub-networks) using at most $uhn(T, 1)$ HSMs on the borderline between the non-upgraded and upgraded sub-networks.

The correctness of the optimal strategy for the HSM upgrading problem also derives from the one of [3]. In particular, from what is stated above we can limit the upgrading to strategies having a single entry point, thus one of the starting point requiring the smallest number of HSMs will be chosen. Another key point is the fact that, given a tree T rooted at r (call it T_r) and considering a sub-tree of T_r rooted at x ($T_r[x]$) with k children $x_1, \ldots, x_k$ such that $uhn(T_r[x_i], 1) \geq uhn(T_r[x_{i+1}], 1)$ for all $i = 1, \ldots, k$, then $uhn(T_r[x], 1) = \max\{uhn(T_r[x_1], 1), uhn(T_r[x_2], 1)+1\}$. Obviously $uhn(T_r[x], 1) \geq uhn(T_r[x_1], 1)$ otherwise $T_r[x_1]$ cannot be upgraded. We now have two possible cases:

1. $uhn(T_r[x_1], 1) \geq uhn(T_r[x_2], 1)+1$. Then $T_r[x]$ is upgraded by visiting $T_r[x_1]$ as the last subtree, after having placed an HSM on x and cleaned all the other children;

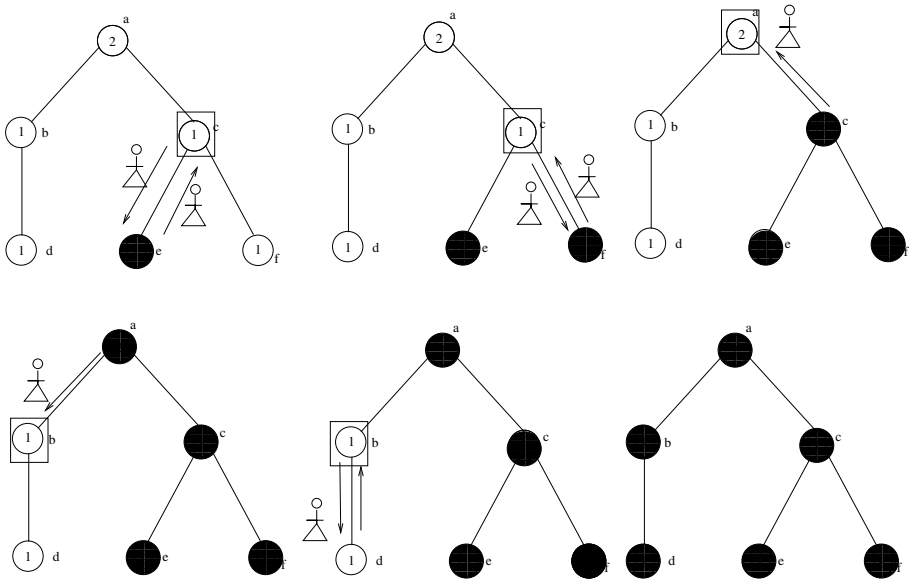

Fig. 5. Algorithm UPGRADE starting from c.

2. $uhn(T_r[x_1], 1) = uhn(T_r[x_2], 1)$. Assume S is a strategy that upgrades $T_r[x]$ using $uhn(T_r[x_2], 1)$ HSMs. If $T_r[x_2]$ is upgraded before $T_r[x_1]$ no HSM may be left on x, thus an untraversed edge (towards x_1) remains unprotected. The same holds if $T_r[x_2]$ is upgraded after $T_r[x_1]$. Thus more than $uhn(T_r[x_2], 1)$ HSMs have to be used. In fact, $uhn(T_r[x_2], 1)+1$ HSMs are sufficient by first visiting $T_r[x_2]$ and then $T_r[x_1]$ with one HSM on x.

Label now each edge $\{a, b\}$ of the network with a function $\lambda_a(a, b)$ that is 0 if b is a leaf and $max\{l_1, l_2+1, 1\}$ if b has h out-going edges each respectively requiring $l_1, \ldots, l_h$ HSMs, with $l_1 \geq l_2 \geq, \ldots, \geq l_h$. The next step borrowed from [3] is to prove that $\lambda_a(a, b) = uhn(T_a[b], 1)$ and this is done by induction on the height of $T_a[b]$.

All the proofs of [3] can thus be directly mapped to our problem. There are some differences on the rules which we report below. Rule 1.1 for the HSM upgrading problem avoids the placement of a borderline switch (i.e., the agent in the other problem) on y, i.e., at the end of the chain, but, by the recursive construction, a borderline switch will however be placed on x while upgrading y, with the only exception being a tree composed of a single edge $\{x, y\}$, that we do not treat (it could be treated via an ad-hoc rule but we prefer not to complicate the algorithm for a trivial case). Now, if x has degree at least 2, by Rule 2 node x will require at least one borderline HSM (used on x) and one technician (the two edges out-going from x have associated 0). Thus, y will be safely upgraded. Moreover, Rule 1.2 states that, if x is the starting point of a chain of HSMs we need at least one borderline HSM to upgrade the system. This is included in the

constant 1 of the relation $\lambda_x(x,y) = \max\{l_1, l_2 + 1, 1\}$. This is peculiar of our algorithm and is not present in [3].

Consider now Algorithm UPGRADE. In point 1 we start from the node that requires the minimal number $B = uhn(T, 1)$ of borderline HSMs. B has been correctly computed by Rule 1 and 2. Consider now point 2 of Algorithm UPGRADE. The proof is similar to the one of [3] with the inclusion of the case where a node is a leaf that has to be directly updated by the technician (that is in this case $\lambda_x(x,y) = 0$ no borderline HSMs, only the technician). Very briefly, the main idea is that technician and HSM move may be forward and backward. Backward moves are safe as they leave an upgraded network and move to an already upgraded one. Forward moves upgrade a non-upgraded subtree by increasing number of HSMs. Moreover the relation $\lambda_x(x,y) = \max\{l_1, l_2 + 1, 1\}$ assures that at least one HSM will be left on node x, avoiding new attacks and enough HSMs can be moved to y.

Finally, observe that Rule 1 and 2 compute the minimal number of borderline switches required on the tree and starting from every possible node, and B is the minimal of such values. ∎

Case U > 1. We now assume that our network is a tree T, and that $U = k > 1$, i.e., that nodes can be simultaneously upgraded by a team of k technicians. Using this issue we now try to compute what is the related decrease in B. To do this we extend Rule 1 and Algorithm UPGRADE 1 basing our changes on the following observation: given $U = k$, and the tree T, if we have a subtree T_z of T rooted at z and containing $<= k$ nodes, we can then use the team to upgrade T_z in one shot, one technician sent to each node. That is, the team can collaborate to upgrade a contiguous part of T, decreasing the number B of borderline nodes required by Algorithm UPGRADE 1.

In order to do this we have to compute, for each node z of T, the number of nodes in each subtree of T rooted in z (i.e., in T_z). More precisely, assuming z has h neighbouring nodes $z_1, \ldots, z_h$, we have to compute s_{z,z_i} the size of the subtrees $T_z[z_i]$ of T_z rooted in z_i, for $i = 1, \ldots, h$. We can do this recursively as follows:

Computing the number of nodes of subtrees

1. An edge $e = \{x, y\}$ leading from a node x to a leaf y. We trivially have $s_{x,y} = 1$;
2. Consider an edge $e = \{x, y\}$ leading from a node x to a node y that has other k out-going edges $\{y, z_1\}, \ldots, \{y, z_k\}$ with subtree sizes s_{y,z_i}, with $i = 1, \ldots, k$. Then the size of the subtree of T_x rooted at y is $s_{x,y} = s_{y,z_1} + \ldots + s_{y,z_k} + 1$.

Based on this computation we can give a new rules, RULE 1 NEW which substitutes Rule 1:

Rule 1 new for computing minimal number of borderline HSMs on an edge using U technicians

1. An edge $e = \{x, y\}$ leading from x to y such that $s_{x,y} \leq U$ requires only the technician team moving from x to y to upgrade the HSM on the whole subtree rooted in y, thus $\lambda_x(x, y) = 0$;
2. An edge $e = \{x, y\}$ leading from x to y such that $s_{x,y} > U$ and having other k out-going edges requiring $l_1, \ldots, l_k$ borderline HSMs, with $l_1 \geq l_2 \geq, \ldots, \geq l_k$, requires $\lambda_x(x, y) = \max\{l_1, l_2 + 1, 1\}$ borderline HSMs.

Theorem 3. *Algorithm* UPGRADE *together with Rule 1 new and Rule 2 correctly solve the HSM upgrading problem in a bank tree network T where U technicians operate. The number of borderline switches required is value B computed via Rule 1 new and Rule 2.*

Proof. Rule 2 and the UPGRADE algorithm are the one previously presented, their correctness thus follows. The correctness of Rule 1 new derives from the one of Rule 1. Moreover, the computation of the sub-tree sizes is based on the standard saturation technique widely used in the field of distributed algorithms (see, e.g., [19]). This technique assumes that the computation of the size of a sub-tree is started at the leaves (which in this case count one) and is propagated on the sub-tree by collecting values from all but one edge and propagating it to through the remaining edge. Finally, given U technicians that may work on a non-upgraded network of $\leq U$ nodes, all these technicians may obviously upgrade this sub-tree in parallel. ∎

4 Estimating the upgrading cost: an example

We consider a simple example to show how to compute a trade-off between the number U of technicians and the the number of needed borderline HSMs. The example is depicted in figure 6. We use the same notation adopted in the previous section to label edges and nodes. On the left, we execute the algorithm with $U = 1$: it gives 2 on all nodes, thus the minimal number of needed borderline HSMs is 2. We now want to evaluate the benefits of hiring one more technician. On the right we have executed the algorithm with $U = 2$, pointing out the difference in red, and we see that in 2 nodes just one HSM is needed. Thus, if we start from those nodes with two technicians we only need one borderline HSM to upgrade the whole network. Notice that we cannot do best than this as one HSM is for sure needed.

We can now reason as follows: let C_H be the cost for one HSM and C_U the cost for one technician. In the first case the overall upgrade cost is estimated as $2C_H + C_U$ while in the second case we estimate as $C_H + 2C_U$. It is now clear that depending on how C_H and C_U are related we will go one direction or the other. For example, if a technician costs around 5000$ and an HSM around 10000$, we will opt for the second solution spending 20000$ instead of 25000$.

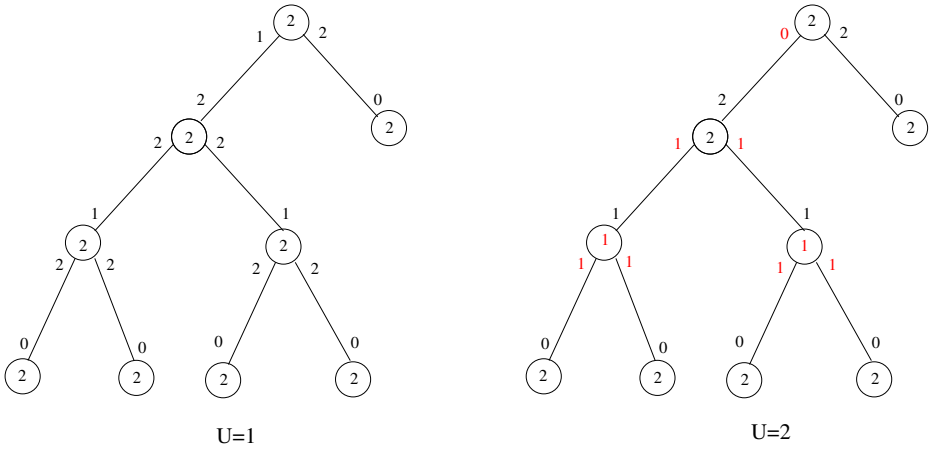

Fig. 6. An example of cost estimation.

5 Conclusion

We have proposed a novel way of upgrading critical components in wide networked system which are incremental and aim at finding a trade-off between the extra hardware needed to maintain the functionality of the network, and the size of the technician teams simultaneously operating on the bank network switches. The presented techniques are not specific for HSMs and PIN managing but could be reused in any networked system requiring the upgrade of some security-critical component.

There are some aspects that we have not treated in this paper and might be interesting to investigate. We have mentioned in section 2 that borderline HSMs might be placed on edges instead of nodes. This increases the number of secured paths but, intuitively, requires more hardware to isolate secure subnetworks from insecure ones. To understand the benefits of this approach it would be useful to measure the degree of security of the network, i.e., the size of the secure subgraph from the ATMs to the related issuing bank or, in other words, the number of secured, upgraded paths in a network. Having this measure, we could compare the present strategies with ones placing borderline HSMs on edges, also comparing the degree of security provided by the new approach. The trade-off, thus, might be between the cost and the degree of security.

Finally, another interesting issue, we have previously mentioned, is the optimization of a new parameter, i.e., the technician motion either inside a weighted network or on a new independent physical network overlapped to the bank network. In this case, time complexity would probably become an issue as in the traveling salesman problem. Depending on the size of the analyzed trees, the solution might thus require heuristic approaches.

References

1. Hackers crack cash machine PIN codes to steal millions. The Times online. http://www.timesonline.co.uk/tol/money/consumer_affairs/article4259009.ece.
2. PIN Crackers Nab Holy Grail of Bank Card Security. Wired Magazine Blog 'Threat Level'. http://blog.wired.com/27bstroke6/2009/04/pins.html.
3. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In *Proceedings of the 14-th ACM Symposium on Parallel Algorithms and Architectures (SPAA),Winnipeg, Manitoba, Canada,*, pages 200–209, 2002.
4. L. Barrière, P. Fraigniaud, N. Santoro, and D.M. Thilikos. Searching is not jumping. In Springer LNCS 2880, editor, *Proceedings of the 29th International Workshop on Graph Theoretic Concepts in Computer Science (WG), Elspeet, the Netherlands*, 2003.
5. O. Berkman and O. M. Ostrovsky. The unbearable lightness of PIN cracking. In Springer LNCS vol.4886/2008, editor, *11th International Conference, Financial Cryptography and Data Security (FC 2007), Scarborough, Trinidad and Tobago*, pages 224–238, February 12-16 2007.
6. M. Bond and P. Zielinski. Decimalization table attacks for pin cracking. Technical Report UCAM-CL-TR-560, University of Cambridge, Computer Laboratory, 2003. http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-560.pdf.
7. R. Breish. An intuitive approach to speleotopology. *Southwestern cavers*, VI(5):72–28, 1967.
8. M. Centenaro, R. Focardi, F. Luccio, and G. Steel. Type-based analysis of PIN processing APIs. In *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS'09)*, pages 53–68. Springer, LNCS 5789, 2009.
9. J. Clulow. The design and analysis of cryptographic APIs for security devices. Master's thesis, University of Natal, Durban, 2003.
10. J.A. Ellis, I.H. Sudborough, and J.S. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113:50–79, 1994.
11. R. Focardi, F. Luccio, and G. Steel. Blunting differential attacks on PIN processing APIs. In *In proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009)*. Springer, LNCS 5838, October 2009.
12. R. Focardi and F.L. Luccio. Cracking bank PINs by playing Mastermind. In *Proceedings of the Fifth International Conference on Fun with algorithms (FUN'10)*. LNCS, Springer, June 2010. To appear.
13. F. Hayashi, R. Sullivan, and S.E. Weiner. *A Guide to the ATM and Debit Card Industry*. Federal Reserve Bank of Kansas City, 2003.
14. L.M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47:205–218, 1986.
15. A. Lapaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, 1993.
16. N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, 1988.
17. T. Parson. Pursuit-evasion problem on a graph. *Theory and applications of graphs*, pages 426–441, 1976.
18. S. Peng, M. Ko, C. Ho, T. Hsu, and C. Tang. Graph searching on chordal graphs. *Algorithmica*, 27:395–426, 2002.
19. N. Santoro. *Design and Analysis of Distributed Algorithms*. John Wiley & Sons, 2006.
20. G. Steel. Formal Analysis of PIN Block Attacks. *Theoretical Computer Science*, 367(1-2):257–270, November 2006.